Windows ドライバー仕様書

（機能、設定、仕様）

Ver. 3.2xx 用

シチズン・システムズ株式会社
東京都西東京市田無町六丁目1番12号
TEL. 0424-68-4993
sales-dp@systems.citizen.co.jp
http://www.citizen-systems.co.jp/

目次

目次 - 1 -
更新履歴(3.xx/2.xx/1.6xx) - 3 -
更新履歴(1.xx) - 4 -
1. オペレーティングシステム - 7 -
2. ドライバーがサポートするモデルとサポートする機能 - 7 -
3. インターフェース - 8 -
4. ドライバーの設定 - 9 -
4．1 印刷詳細設定 - 9 -
4．1．1 用紙サイズの設定 - 10 -
4．1．2 グレースケール(階調)印刷の設定 - 11 -
4．1．3 NV Logo Process の設定 - 13 -
4．1．4 Paper Type - 14 -
4．1．5 PaperMedia - 14 -
4．1．6 PaperFeed - 15 -
4．1．7 CutterMode - 15 -
4．1．8 Cash Drawer #1, #2 - 16 -
4．1．9 Buzzer - 17 -
4．1．10 Bar Code Printing - 17 -
4．1．11 Logo Printing - 18 -
4．2 拡張機能の有効無効設定 - 20 -
4．3 電子ジャーナル - 1 -
4．3．1 ジャーナルビューワ - 5 -
4．4 透かし印字 - 6 -
4．5 上下反転印刷 - 8 -
4．6 クーポン印刷 - 10 -
4．7 ファイル送信 - 13 -
4．8 POS Printer Utility - 14 -
4．9 再印刷 - 15 -
4．10 ドライバーポート設定 - 17 -
4．11 ステータスモニターライブラリー - 18 -
4．12 バージョン表示 - 19 -
5. プリンターフォント - 20 -
6. バーコード印刷 - 23 -
7. 二次元バーコード印刷 - 24 -
7．1 二次元バーコードフォント書式 - 24 -
7．2 二次元バーコードフォントの使用例 - 25 -
8. グラフィック印刷 - 27 -
9. 特殊機能 - 28 -
10. 用紙サイズ - 29 -
11. プリンターステータス - 33 -
11．1 プリンターステータスの取得 - 33 -

11. 2 双方向通信の有効／無効 ..... - 35 -
12. 特定アプリケーションにおける使用例 ..... - 37 -
12. 1 Microsoft Word での使用例 ..... - 37 -
12. 2 Visual Basic でのプログラム例 ..... - 38 -
12. 3 Visual C++でのプログラミング例 ..... - 39 -
12. 4 Visual Basic.Net でのプログラミング例 ..... - 41 -
12. 5 Visual C#.Net でのプログラミング例 ..... - 46 -

更新履歴(3.xx/2.xx/1.6xx)

年月日	バージョン	履歴
09/06/25	V2.00	発行
09/10/29	V2.01	CT-S601 を追加
09/12/15	V2.02	CT-S281、CT-S4000 を追加 Windows7 対応 ステータスモニター付ドライバーを標準とする ドライバーポートの設定を追加 ステータスモニターライブラリーを追加
10/05/20	V2.02 (修正版)	CT-S651/851 を追加
10/5/27	V2.23	バージョンの付け方をドライバー本体のバージョンに合わせた。 ステータスモニターを Ver2.2.4.0 へ更新しオンライン／オフライン状態が取得できるようになった。 ステータスモニターのデフォルト USB 送信タイムアウト値を 8 秒へ変更し Windows7 の USB 経由で双方向ドライバーを使用すると通信エラーになる問題に対応。 各 dll の FileDescription に 32-bit/64-bit の表記を追加。 CT-S601、651、801、851 の Windows7 デバイスアイコンを追加。
11/1/27	V2.24	サポートするインターフェースにイーサネットを追加 プリンターステータスの説明の追加 Visual C++でのプログラミング例のソースコード修正 POS Printer Utility を Ver1.8.3 へ更新。(設定値の間違い修正) Status Monitor Library を Ver1.3.1 へ更新。(ステータス取得時間を改善) ブラックマーク用のドライバーのドライバー名を変更 CITIZEN CT-S651 Label → CITIZEN CT-S651 Black Mark CITIZEN CT-S851 Label → CITIZEN CT-S851 Black Mark Printer Functions の"Label Paper"→"Label/BM"へ名称変更 NV Logo Process のデフォルト値を FS q → GS (L へ変更 (CT-S601,651,801,851,2000,4000,PPU-700) ステータスモニター、スプーラーリセット時、電源オン時に双方向通信が停止されていたら、再開するように変更。
11/4/13	V2.25/1.660	二次元バーコードフォントを追加。 "Label/BM"機能を"Paper Media"機能へと変更。 "Cutter Mode"を変更、Label/BM Paper のカット動作を選択可能にした "Paper Feed"機能を新規追加し、カット前の紙送り量が調整可能にした ファイル送信機能のゴミ印字の不具合に対応 ボタンとタブが環境により可視、不可視になる機能を追加 ステータスモニターを 2.2.5.2 にマイナーチェンジ POS プリンターユーティリティーをドライバーインストーラーからはインストールされなくなった POS プリンターユーティリティー2.0 にアップデート 説明文書に使われる画像を Windows7 のものに変更 Ver1.660 の説明を統合
12/6/15		TCP/IP ポート の説明を追加
13/3/1	V2.27/1.680	Windows8 対応 バージョンダイアログの追加 ウォーターマークの途中でカットが入る不具合を修正 クライアント PC から共有プリンターで再印刷、クーポン印刷が出来ない不具合を修正 ステータスモニターを 2.2.5.3 に更新 Win8/Win Server2012 に対応 ログファイル機能の改良 オフラインステータスの取得有り無しの設定を追加 印刷完了通知機能の有効無効を設定可能にした パラレル以外での I/F 用問合せコマンドの内容の変更 ステータスモニターの設定をドライバー単位からモデル単位に変更 ドライバーポート設定ツールを更新
13/7/22	V2.28/V1.690	独語、仏語版などの Windows8 において、インストールが失敗する不具合に対応しました。 LAN、WLAN との組合せで、プリンターのエラーステータスが取得出来ない事がある不具合に対応しました。 CITIZEN PMU2xxxIII Presenter のステータスの名前判定の間違いを修正しました。 ドライバーのアンインストール画面のアイコンから"X"マークを削りました。
14/1/21	V2.281/1691	CT-S281BD のサポートを追加した。
14/10/15	V3.00	サポート OS から Windows2000 を削除 CT-S251/S8xxII/6xxII の追加 CT-S251/S8xxII/6xxII 用の階調ロゴ、階調透かし印字機能を追加
14/12/15	V3.100	「4.1.2 グレースケール(階調)印刷の設定」を追加
15/02/27	V3.101	ユーザ定義用紙で横方向グレースケール印刷出来ない不具合を修正
15/08/20	V3.200	Windows10 に対応 拡張機能の有効無効切替機能の追加により、1.xx/2.xx 系のドライバーを 3.xx 系に統合

更新履歴(1.xx)

年月日	バージョン	履歴
05/04/18	050418	リリース候補版 CD-S500 シリーズに日本語フォント追加 フォントの大きさの説明、2色印字の説明を変更
05/04/20	050420 （V1.00）	ラベル、BM機能を追記
05/05/18		誤記訂正(日本語フォント A と C 入れ替え)
05/09/01	050901 （V1.20）	CT-S280 用ドライバー追加 CD-S500 ドライバーに FontAxx / FontBxx を追加した。 フル桁より長いデータ印字時に印字が乱れる問題を修正 15cpi フォント サイズ 80 にてフル桁印字できない問題を修正 プリンターフォントと TrueType フォントの混在による印字の乱れの問題を修正 CBM1000II で“漢字 3.8cpi”+サイズ 24 のデータが印字されない問題を修正 ステータス機能を追加 CT-S300 のカスタム用紙サイズでの印字乱れ問題を修正
05/11/15	051115 （V1.51）	CT-S2000 を追加
06/01/28	V1.53	パラレルでグラフィックが文字に化ける不具合を修正 CD-S500 の高品質グラフィックが乱れる不具合を修正 CD-S500 の桁あふれ時の不具合を修正
06/07/30	V1.56	CT-S280 のページ末に自動改行を追加 バーコード印字後の改行量の不具合を修正 グラフィック印字中の改行量を 0 に変更 上記の変更で落ちた印字スピードを改善 CT-S300 のグラフィック方式を変更 CT-S4000 を追加
07/3/20	V1.57	バーコード印字後の改行量の不具合（その 2)を修正 文字サイズが不安定になる不具合を修正 CT-S4000 にラベル機能を追加 BD2-2220/2221 を追加 「プリンターの機能」項目を追加、固定長印字に対応した Windows Vista に対応
07/4/30	V1.57a	CT-S310 を追加 PMU2xxx を追加
07/12/21	V1.581	64 ビットドライバーを追加 インストーラーに対応 CT-S2000Label を追加 CT-S4000 の圧縮ドライバーを追加 PMU-2xxx のカット方法にパーシャルカットを追加した。 CT-S2000/4000/PPU700 で ESC (L によるロゴ登録が選べるようにした。 CT-S280 のイメージ処理方式をラスター方式からカラム方式へ変更した。 Unicode で一部の文字が印字できない問題を修正 日本語フォント B 追加(CT-S2000/CT-S4000)。 日本語フォント指定時の半角文字の横幅情報の間違いを修正。 日本語フォント A/B/C 指定時、同時に FontA/B/C を指定出来るように変更した。 FontA88 の横幅情報の間違いを修正。 6.25cpi 48pt の横幅情報の間違いを修正。 Logo Print 機能設定で複数部数を発行する場合、設定した箇所に指定 Logo が 正しく印字されなかった不具合を修正。 ページ終端位置にあるプリンターフォントが正しく印刷されない問題を修正。 倒立印字(Upside Down Printing)を複数部行った際の用紙長の問題を修正
08/07/02	1.582	ドライバの[Tool]プロパティより POS Printer Utility を起動出来るようになりました。 ドライバの[Tool]プロパティにファイル転送機能を追加しました。 UpsideDown 印刷有効時にプリンタフォントの印字位置がおかしくなる不具合を修正。 CD-S500 の高解像度のイメージ印刷時、最後の縦ドットラインが欠けてしまう不具合を修正。 CD-S500 系の用紙幅設定で実印刷領域を拡大。 横方向印刷時に白線が入ってしまう問題に対応。（この不具合対応はファームウェアの変更と併せて有効になります。対応機種は CT-S2000、CT-S4000、CT-S310 と今後のモデルのみ。） CT-S300、310 のカッターコマンドを GS V から ESC i, ESC m に変更しました。 CT-S2000/4000/S28x/BD2-2xxx/PMU-2xxx/CBM1000II の FontBxx を正しいサイズに修正した。 CT-S281 のドライバーを追加。 ラベルドライバでページ末に配置されているデータを印字出来なかった問題を修正
08/12/25	1.582a	横方向印刷時白線問題への対応が CT-S310 の旧ファームウェアでゴミ印字を起こすため、CT-S310 用ドライバーでは白線問題対応を削った。

09/04/08	1.583	CT-S281L のドライバーを追加 全機種の GPD ファイル内のカスタム用紙定義を以下のように修正した。 ・左マージン(*MinLeftMargin)の削除：全機種 ・最大定義サイズ(*MaxSize)を修正：CT-S4000 系(640,32767)→(896,26182)、その他機種(???,32767)→(???,26182) ※CT-S500 系は非該当。 ・最小定義サイズ(*MinSize)を変更：ラベル系→(203,203)、非ラベル系→(20,20)、PPU-700→(203,719) ※CD-S500 系と CT-S4000 の圧縮ドライバは変更無し CT-S281 の標準ドライバーで横方向白線問題の対応をした。 ローカルプリンタ名(「プリンタと FAX」の所の名前)の変更で、POS Printer Utility 起動ボタンが消えてしまう問題に対応。
09/12/15	1．600	CT-P290 シリーズに対応しました。 リモートデスクトップからの印刷が出来ない問題に対応しました。 Windows7 上で NV ロゴ印刷の順番がおかしくなる問題に対応しました。 "Printer functions"設定ツリーのアイコンを変更しました。 上下反転印刷機能を削除しました。 ラベル・ブラックマーク用ドライバーを標準ドライバーに統合しました。
10/2/9	1.610 （未公開）	ステータスモニターを一新し、ステータス機能が大幅に強化されました。
10/6/21	1.630	複数プロセスからの印刷による不具合を修正。 印刷の最後にロゴ印刷指定＋直前の印刷が縦倍のプリンタ内蔵フォントという条件の組み合わせで、ロゴ登録の有無で改行量が変わってしまう不具合を修正。 各 DLL の FileDescription へ (32-bit)/(64-bit) 表記を追加修正。 ステータスモニターが Ver2.2.4.0 へ更新され、オンライン／オフライン状態が取得できるようになった。 ステータスモニターのデフォルト USB 送信タイムアウト値を 8 秒へ変更し Windows7 の USB 経由で双方向ドライバーを使用すると通信エラーになる問題に対応。 PPU-700 のプレゼンター出口での用紙検出を PRINTER_STATUS_OUTPUT_BIN_FULL として対応。
11/1/6	1.640	BD2-4280,BD2-4281 をサポートしました。 PMU-2320 ドライバーを PMU2xxx ドライバーと名称を変更しました。 PMU2xxxIII Presenter をサポートしました。 Status Monitor Library を Ver1.3.1 へ更新。（ステータス取得時間を改善。）
11/4/13	1.650	BD2-4280,BD2-4281 をまとめ、BD2-428x としました。 二次元バーコードフォントを追加。 "Label/BM"機能を"Paper Media"機能へと変更。 "Cutter Mode"を変更、Label/BM Paper のカット動作を選択可能にした "Paper Feed"機能を新規追加し、カット前の紙送り量が調整可能にした ファイル送信機能のゴミ印字の不具合に対応 ボタンとタブが環境により可視、不可視になる機能を追加 ステータスモニターを 2.2.5.2 にマイナーチェンジ POS プリンターユーティリティー2.0 にアップデート

ご注意

(1) 本書の内容の一部、または全部を無断で転載することは、固くお断りいたします。

(2) 本書の内容については、事前の予告なしに変更することがあります。

(3) 本書の内容については万全を期して作成いたしましたが、万一誤り・お気付きの点がございましたら、ご連絡くださいますようお願いいたします。

(4) 運用した結果の影響につきましては、(3)項にかかわらず責任を負いかねますのでご了承ください。

(5) 上記に同意いただけない場合は、本ドライバーをご使用いただけません。

商標

Windows XP、Windows Server 2003、Windows Vista、Windows 7、Microsoft Windows 8、Microsoft Windows 8.1、Windows10、Visual Basic、Visual C++、Visual C#、.Net、Microsoft Word、Microsoft Access、TrueType は米国マイクロソフト社の登録商標です。

その他、記載されている会社名、製品名は、各社の商標または登録商標です。

CITIZEN は、シチズンホールディングス株式会社の登録商標です。
CITIZEN ⓒ シチズン・システムズ株式会社 2006 年

1. オペレーティングシステム

このドライバーは以下のシステムに対応しております。

Microsoft Windows XP

Microsoft Windows Server 2003

Microsoft Windows Vista

Microsoft Windows 7

Microsoft Windows 8, 8.1

Microsoft Windows10

サーバーOS など、上記をベースとした派生 OS

2. ドライバーがサポートするモデルとサポートする機能

各モデルには機能に違いがあり、ドライバーもそれに合わせて機能に違いがあります。

	ドライバー拡張機能				ファイル送信	双方向通信	
	グレースケール印刷	透かし印刷		電子ジャーナル／上下反転印刷 クーポン印刷／再印刷		ドライバポート設定	ステータスモニターライブラリー
		NV ロゴ選択式	画像選択式				
CT-S251 CT-S601II CT-S651II CT-S801II CT-S851II	✔	✔	✔	✔	✔	✔	✔
CT-S2000 CT-S281 CT-S310 CT-S4000 CT-S601 CT-S651 CT-S801 CT-S851			✔	✔	✔	✔	✔
CT-P29x PPU-700II			✔	✔	✔	✔	✔
CT-S280 CT-S300 PPU-700			✔	✔	✔		
PMU2xxx PMU2xxxIII Presenter					✔	✔	✔
BD2-222x BD2-428x CBM1000II CD-S500 CD-S501/503					✔		

ドライバー拡張機能は、設定により無効にすることが出来ます。

なお、以降の説明で(II)とあるものは、タイプ I とタイプ II の両方を指します。

ドライバーの種類としては、派生モデル、派生機能用に以下のようなドライバーも用意されています。

CT-S281 Label

CT-S310 Black Mark

CT-S2000 Label

CT-S4000 Label

CT-S4000 Compress

CT-S801 Label

CT-S801 II Label

CT-S851 II Black Mark

CT-S851 Black mark

CT-S651 II Black Mark

CT-S651 Black mark

3. インターフェース

サポートするインターフェースは以下の通りです。

シリアル／CITIZEN COM／Bluetooth SPP での接続により生成された COM

パラレル

USB

USB 仮想 COM

LAN、無線 LAN／CITIZEN TCP/IP

（本書は、LAN、無線 LAN インターフェース用の TCP/IP ポートモニターの説明を含んでおりません。
別途 TCP/IP ポートマニュアルを参照ください。）

ご注意

USB 仮想 COM ドライバー

CT-S251/CT-S601(II)/CT-S651(II)/CT-S801(II)/CT-S851(II)を USB 仮想 COM としてお使いになる場合は、USB 仮想 COM ドライバーをインストールする必要があります。

CITIZEN COM ポートモニター

シリアルインターフェースを使用し、プリンターの状態を監視するには CITIZEN COM ポートモニターが必要です。1.6xドライバーでは CITIZEN COM ポートモニターのインストールを事前に行ってください。

CITIZEN TCP/IP ポートモニター

LAN、無線 LAN インターフェースを使用される場合は、プリンターの状態を監視するために、CITIZEN TCP/IP ポートモニターをご使用下さい。

プリンターが複数の PC から LAN または無線 LAN で接続されている場合に、その中のどれかの PC が LAN 接続を終了せずに電源 OFF 等した場合、プリンターは通信が途切れていることを確認し、その PC との LAN 接続を止めますが、確認をする間、他の PC との通信が止まります。（最長 180 秒）

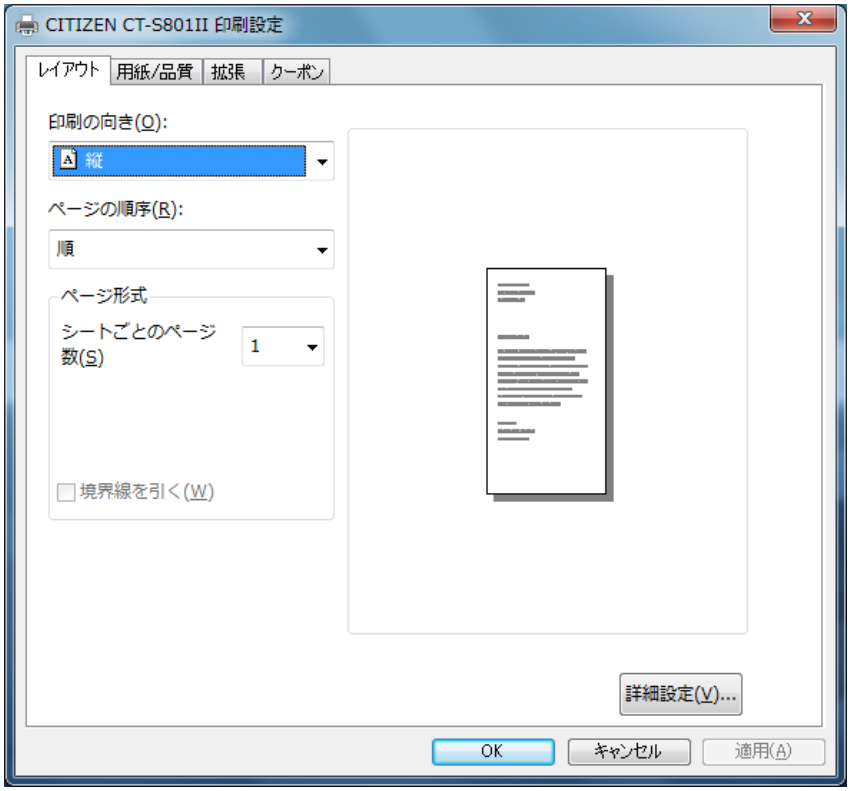
4. ドライバーの設定

ドライバーの機能を設定する方法を説明いたします。

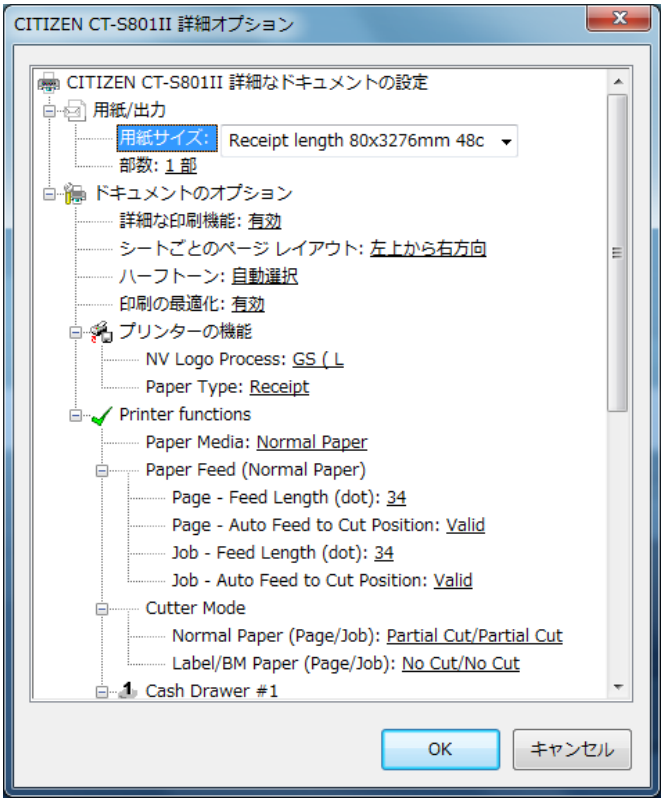
4. 1 印刷詳細設定

プリンタードライバーのプロパティ画面より、「基本設定」を選びます。

右のような画面が表示されますので、「詳細設定」を選びます。



右のような画面が表示されます。印刷詳細設定はこの画面上で行います。

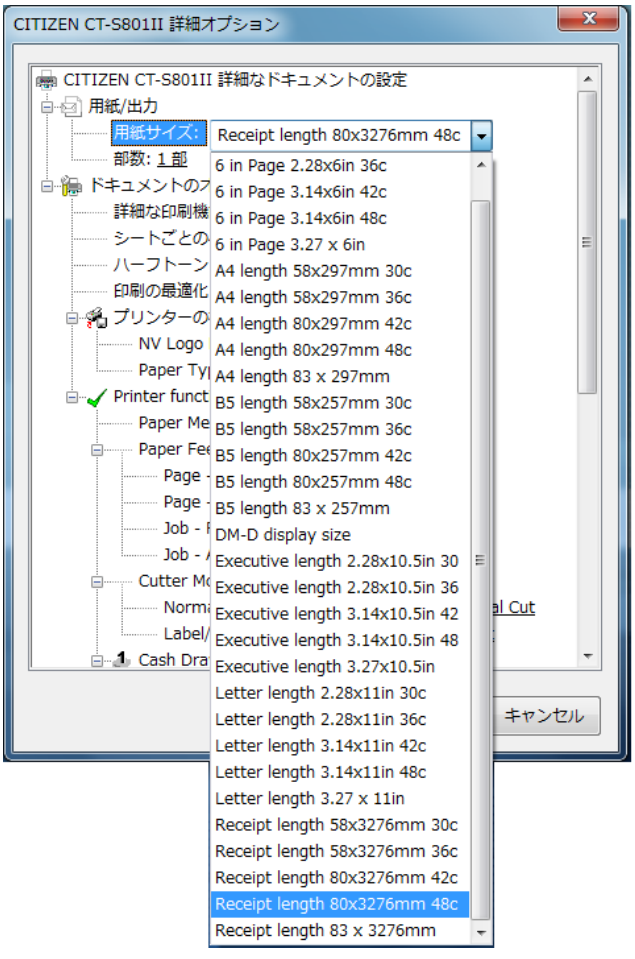


4．1．1 用紙サイズの設定

用紙サイズを選択します。設定出来る用紙はプリンターによって変わります。

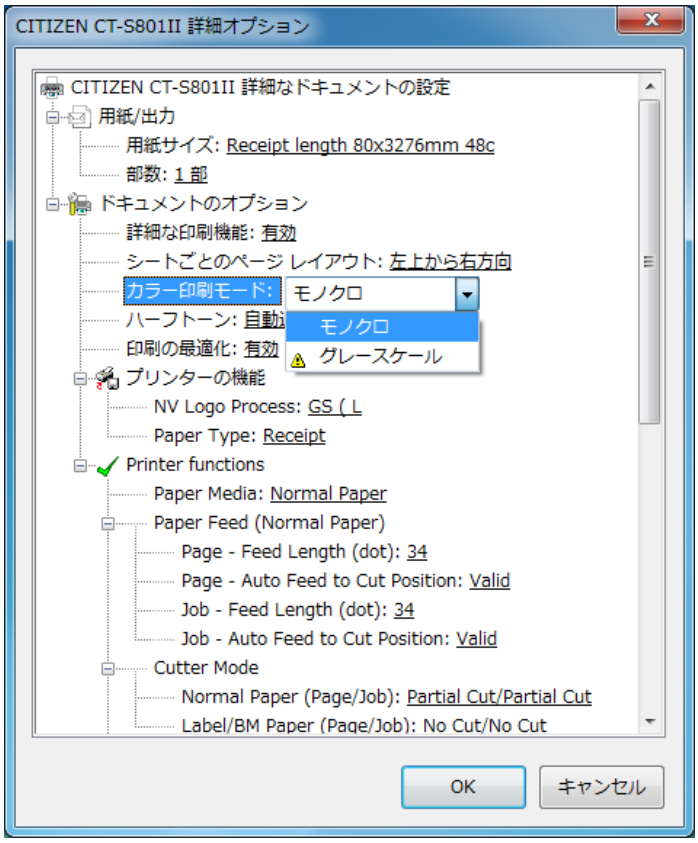
デフォルトはプリンタードライバーの機種によって異なります。

※ユーザー設定用紙も選択できます。詳細は後述の章「用紙サイズ」をご覧下さい。



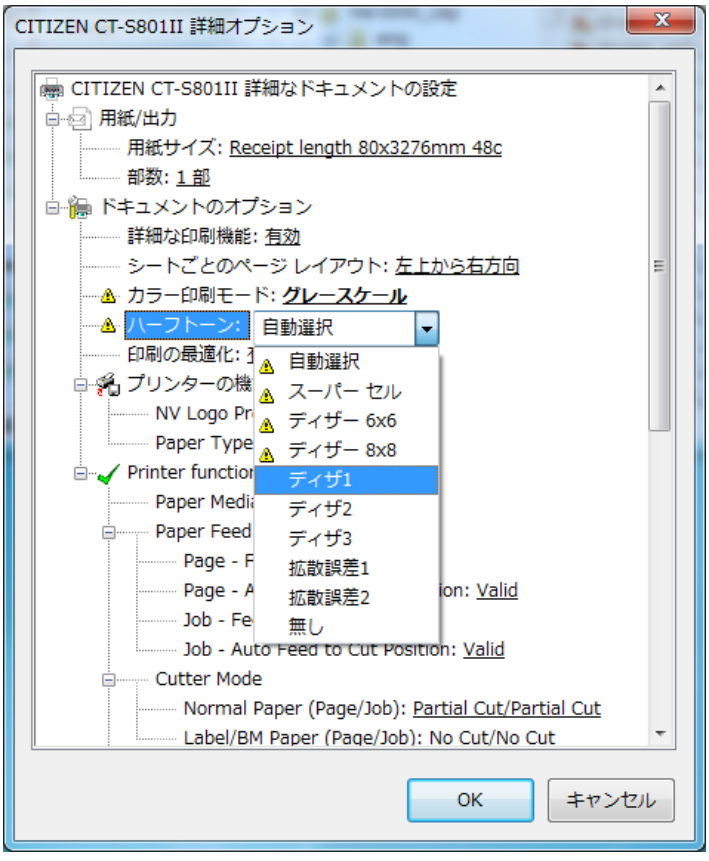
4．1．2 グレースケール(階調)印刷の設定

「カラー印刷モード」の箇所で「グレースケール」を選択すると、グレースケール(階調)印刷が行えます。



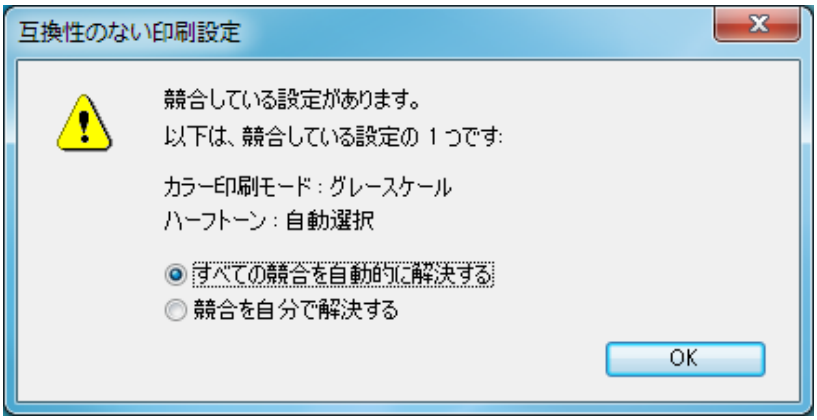
その直ぐ下にある「ハーフトーン」の箇所でディザリング形式を選択出来ます。以下の 6 通りが用意されています。

- ディザ 1
- ディザ 2
- ディザ 3
- 拡散誤差 1
- 拡散誤差 2
- 無し



「グレースケール」と「モノクロ」では選択出来るディザリング形式が異なります。間違った組み合わせを選んだ場合は！マークが表示されますので、選んだ選択肢を見直して下さい。

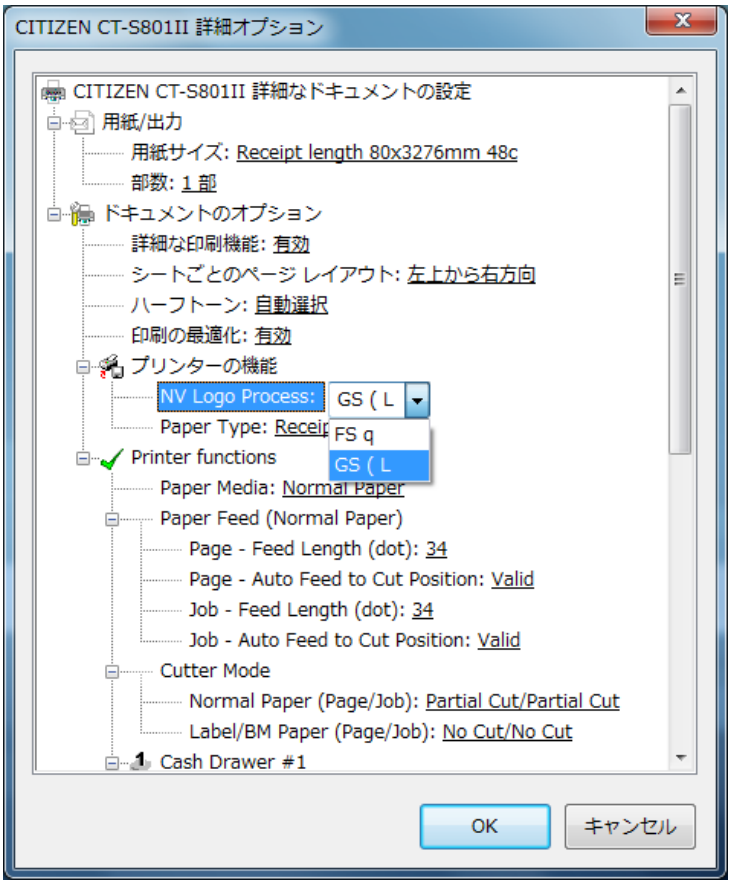
！マークのまま「適用」ボタンを押しても右のメッセージが出て、適切な組み合わせへと自動的に修正されます。



4．1．3 NV Logo Process の設定

プリンター本体に登録されている NV ロゴの書式を選択します。「4．1．10 Logo Printing」を使用する際に、ここで選択された方式の NV ロゴコマンドが採用されます。

デフォルト設定値はそれぞれのプリンタードライバー毎に、CITIZEN POS Printer Utility を使用して NV ロゴ登録する書式側へ設定されています。



FS q の NV ロゴ登録／削除は、一括登録／一括削除方式で、登録されたロゴも、登録順の番号で扱います。それに対し、GS (L の NV ロゴ登録／削除は、個別登録／個別または一括削除方式で、登録するロゴには 2 文字のキーコードを付ける事が出来、このキーコードでアクセスする事が出来ます。

NV ロゴのグレースケール（階調）印刷を行う場合、GS（L の NV ロゴを選択して下さい。グレースケール（階調）の NV ロゴ登録は POS Printer Utility を使用して下さい。

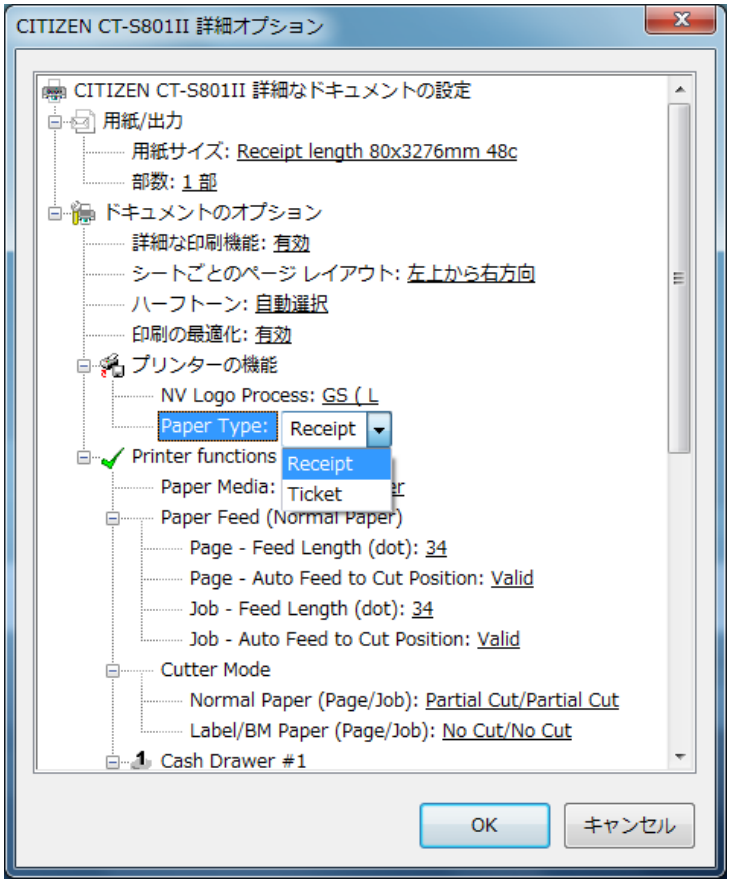
GS (L 機能はサポートしていないモデルもあります。「Paper Type」が表示されていないモデルは、GS (L をサポートしておらず、FS q のみがサポートされています。
本設定メニューが表示されないプリンタードライバーは FS q の NV ロゴ書式のみのサポートとなります。
GS（L との両書式をサポートしているプリンタードライバーは、CITIZEN POS Printer Utility を使用して NV 登録出来る書式へ、デフォルト設定されています。

4．1．4 Paper Type

用紙タイプを Receipt／Ticket から選択できます。

Receipt は、用紙に無駄が出ないように、印字データに合わせて用紙のサイズが変わります。Ticket は、途中で印字データな無くても、設定された用紙サイズの紙送りが行われます。

デフォルトは Receipt です。

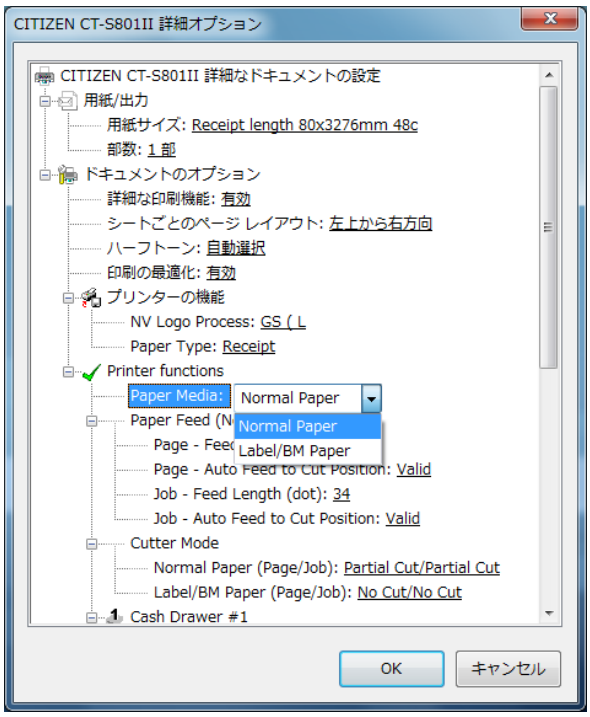


4．1．5 PaperMedia

使用する用紙の種類を選択します。

普通紙、またはラベル紙／ブラックマーク紙へと切り替えられます。

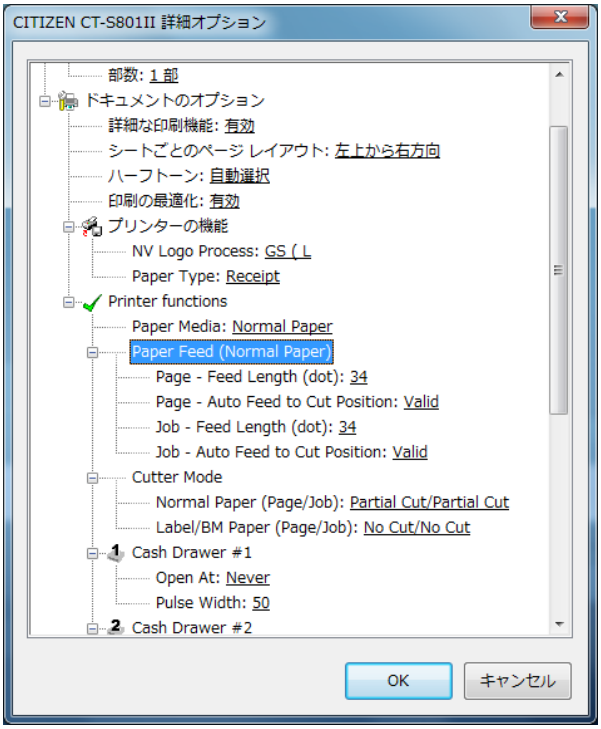
※ラベル紙／ブラックマーク紙をサポートしていないプリンターは、Label/BM Paper は選択出来ません。



・ラベル紙／ブラックマーク紙使用時は、4.1.3 Paper Type を Receipt 側に設定してご使用下さい。

4. 1. 6 PaperFeed

ページ終了部(Page)と、レシート終了部(Job)における用紙送り量とその動作を設定出来ます。



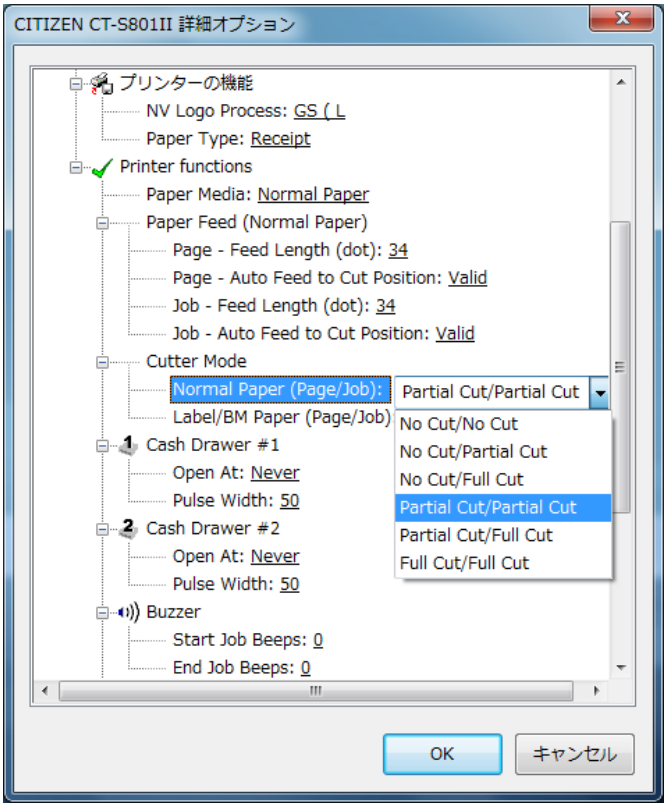
Feed Length (dot)：
0 ～ 406 ドットの範囲で指定できます。

Auto Feed to Cut Position：
プリンター固有のカット位置まで自動紙送りするかどうかを指定出来ます。

・Auto Feed to Cut Position 有効、かつ Feed Length が指定されている場合は、まずカット位置まで自動的に改行され、その後指定されている長さの紙送りが行われます。

4. 1. 7 CutterMode

ページ終了部(Page)と、レシート終了部(Job)の用紙カット動作を設定出来ます。



普通紙使用の場合、プリンタードライバーの機種によってはパーシャルカットとフルカットが選べる物があります。最大で 6 種類の組合せが行えます。

ラベル紙／ブラックマーク紙使用時の場合、カット動作の選択を行えます。

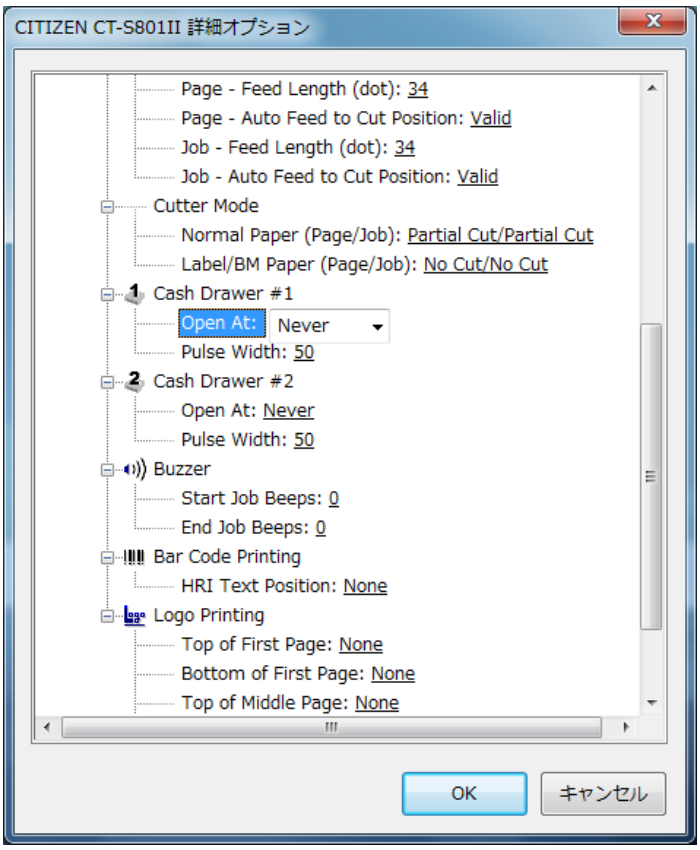
デフォルトはプリンタードライバーの機種によって異なります。

・パーシャルカットとは、用紙の一部分だけを切り残してカットする方式の事を言います。
・オートカッターが搭載されていない機種のプリンタードライバーには、本設定メニューがありません。
・プリンター本体のメモリースイッチ設定に、「強制パーシャルカット」の設定メニューがあります。この設定を有効にしてありますと、ドライバーの Cutter Mode で Full Cut させても、パーシャルカットを行うようになりますので、ご注意下さい。詳細は製品に添付されている取扱説明書の「メモリースイッチのマニュアル設定」の欄をご覧下さい。

4．1．8 Cash Drawer #1, #2

ドロワー1／2 の動作を設定出来ます。ドロワーを開けるタイミング、開ける時のパルス信号長さを、それぞれ設定出来ます。タイミングはレシート開始部(Job Start)、レシート終了部(Job End)、ページ終了部(Page End)から選択出来、パルス信号幅は 50ms～250ms の間で指定出来ます。

デフォルトはドロワー1／2 共、使用しない設定になっています。

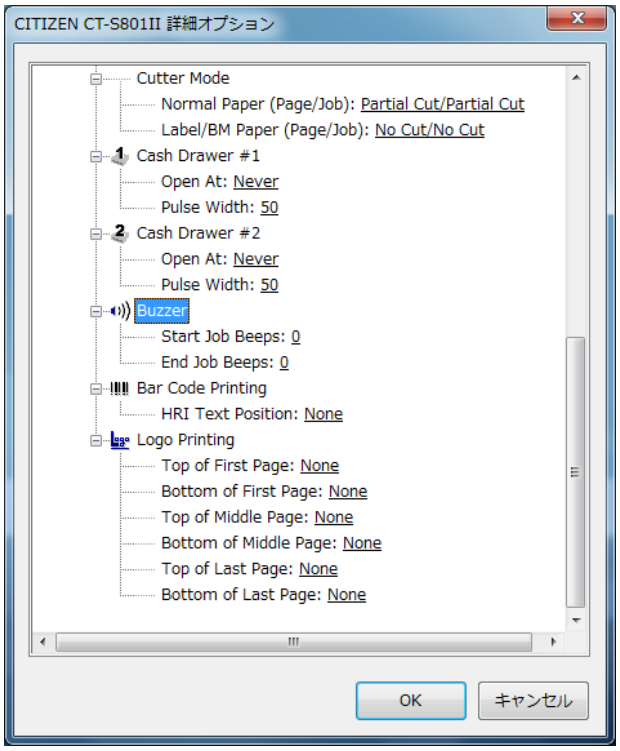


・キャッシュドロワーとはレジで扱う紙幣や貨幣、金券などを一時的に入れておく引出しの事です。
・ドロワーコネクタが搭載されていない機種のプリンタードライバーには、本設定メニューがありません。
・2つのドロワーが接続されている場合、2つのドロワーを同時に開けると、プリンターの故障に繋がる事がありますので、同じタイミングでドロワーを動作させないようにして下さい。
・ドロワーのパルス信号幅のデフォルト値は 50ms に設定されています。ドロワーの種類によっては、このパルス信号幅では引出しが開かない事がありますので、適宜調整下さい。
・プリンターとドロワーの接続について詳細は、製品に添付されている取扱説明書の「ドロワーの接続」の欄をご覧下さい。

4. 1. 9 Buzzer

プリンターブザーの動作を設定出来ます。ブザーを鳴らすタイミングはレシート開始部(Job Start)、レシート終了部(Job End)から選択出来、1～9 回の間で指定出来ます。

デフォルトはブザーを使用しない設定になっています。



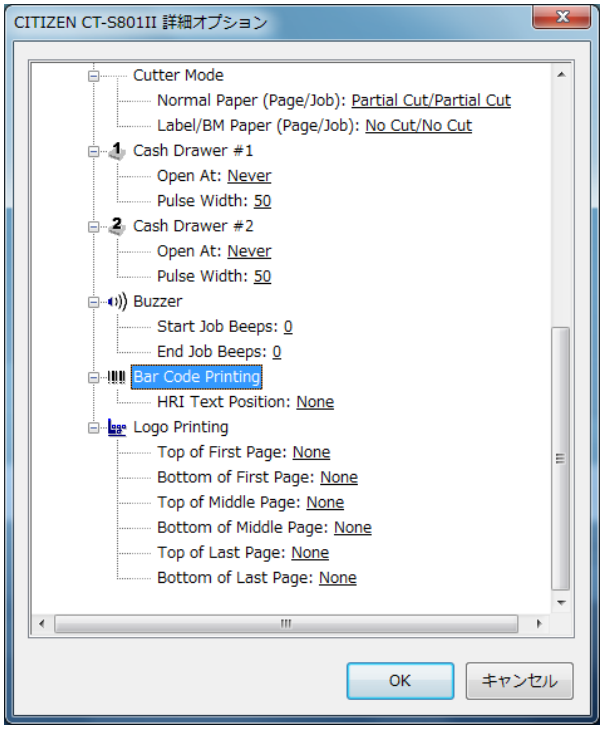
・ブザーが搭載されていない機種のプリンタードライバーには、本設定メニューがありません。

4. 1. 10 Bar Code Printing

バーコード印刷時の可視コードの印刷場所を設定出来ます。上(Top)、下(Bottom)、上下(Top and Bottom)から選択出来ます。

デフォルトは印刷しない設定になっています。

バーコード印刷は後述の章「バーコード印刷」を参照下さい。



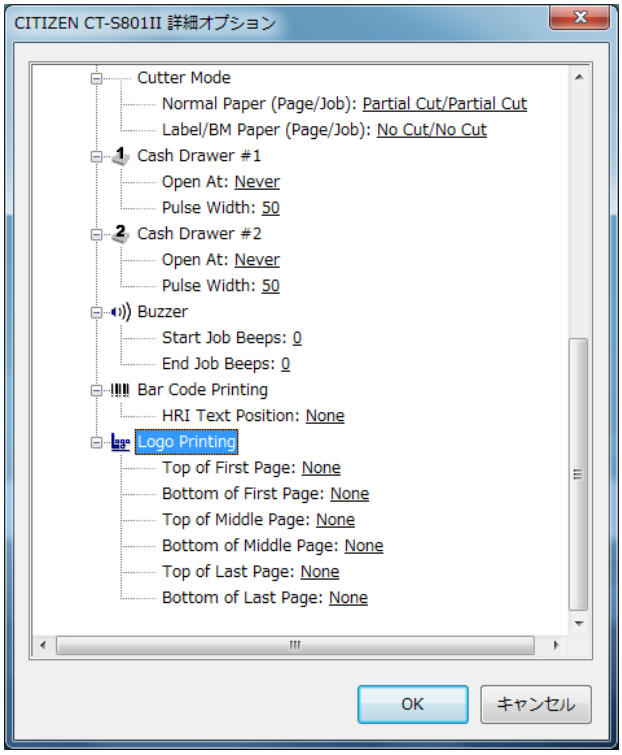
・バーコード印刷がサポートされていない機種のプリンタードライバーには、本設定メニューがありません。

4．1．11 Logo Printing

プリンター本体に登録されている NV ロゴの印刷を設定出来ます。

NV ロゴの印刷位置は、印刷する各ページの初めと終わりの位置が指定出来ます。複数ページ印刷の場合、先頭ページと最終ページにつき、個別の NV ロゴ印刷を指定出来ます。印刷する NV ロゴは logo1～logo9 を指定出来ます。

デフォルトは印刷しない設定になっています。



・「4．1．2 NV Logo Process」で選択された書式の NV ロゴコマンドを使用します。本体に登録されている NV ロゴは FS q コマンドによる物であるか、GS (L コマンドによる物であるかを確認下さい。各コマンドについては「コマンドリファレンス」を参照下さい。

・CITIZEN POS Printer Utility を使用すると、容易に NV ロゴ登録を行えます。

・CITIZEN POS Printer Utility を使用して、GS (L コマンド書式型の NV ロゴを登録される場合、プリンタードライバー連動でロゴを印刷させるには、NV ロゴに予め決められたキーコード「L1」「L2」「L3」「L4」「L5」「L6」「L7」「L8」「L9」のいずれかを付けて登録させる必要があります。

・FS q コマンドによる登録の場合、各ロゴは以下のようになっています。

logo1 ： 一番目に登録された NV ロゴ
logo2 ： 二番目に登録された NV ロゴ
logo3 ： 三番目に登録された NV ロゴ
logo4 ： 四番目に登録された NV ロゴ
logo5 ： 五番目に登録された NV ロゴ
logo6 ： 六番目に登録された NV ロゴ
logo7 ： 七番目に登録された NV ロゴ
logo8 ： 八番目に登録された NV ロゴ
logo9 ： 九番目に登録された NV ロゴ

・GS (L コマンドによる登録の場合、各ロゴとキーコードの関連は以下のようになっています。

logo1 ： キーコード「L1」で登録された NV ロゴ
logo2 ： キーコード「L2」で登録された NV ロゴ

logo3 : キーコード「L3」で登録された NV ロゴ
logo4 : キーコード「L4」で登録された NV ロゴ
logo5 : キーコード「L5」で登録された NV ロゴ
logo6 : キーコード「L6」で登録された NV ロゴ
logo7 : キーコード「L7」で登録された NV ロゴ
logo8 : キーコード「L8」で登録された NV ロゴ
logo9 : キーコード「L9」で登録された NV ロゴ

4．2　拡張機能の有効無効設定

拡張機能を、まとめて無効にすることができます。通常は有効のままでお使いください。

無効にすることで、様々な機能が使えない代わりに、印字速度が若干上がります。

拡張機能に含まれれるのは、以下の機能です。

グレースケール印刷

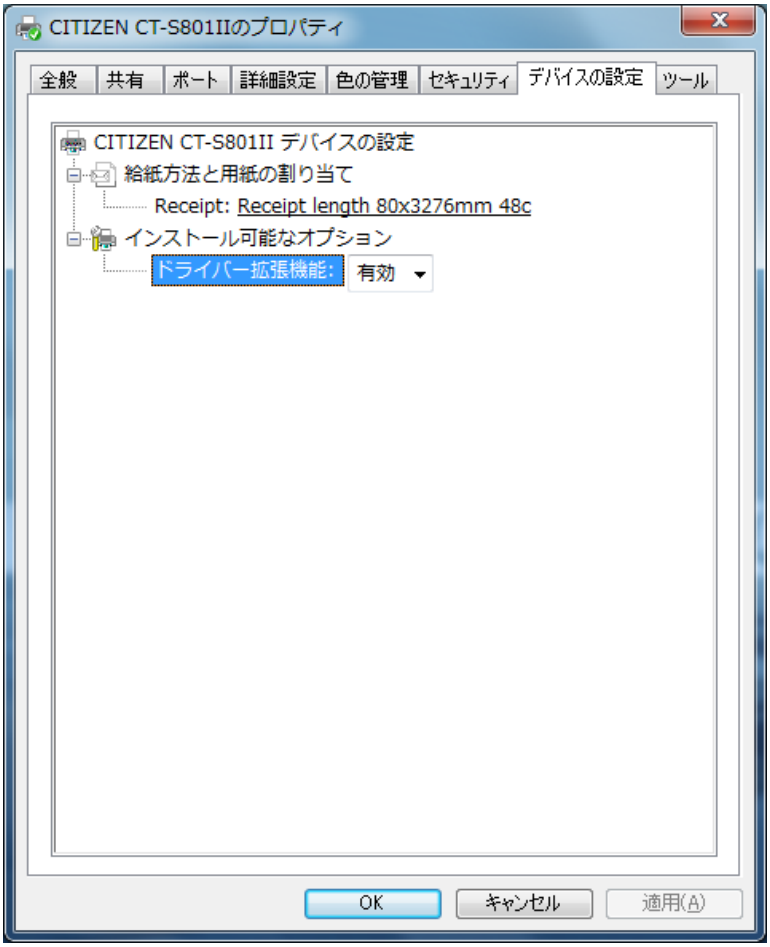
透かし印刷

電子ジャーナル

上下反転印刷

クーポン印刷

再印刷

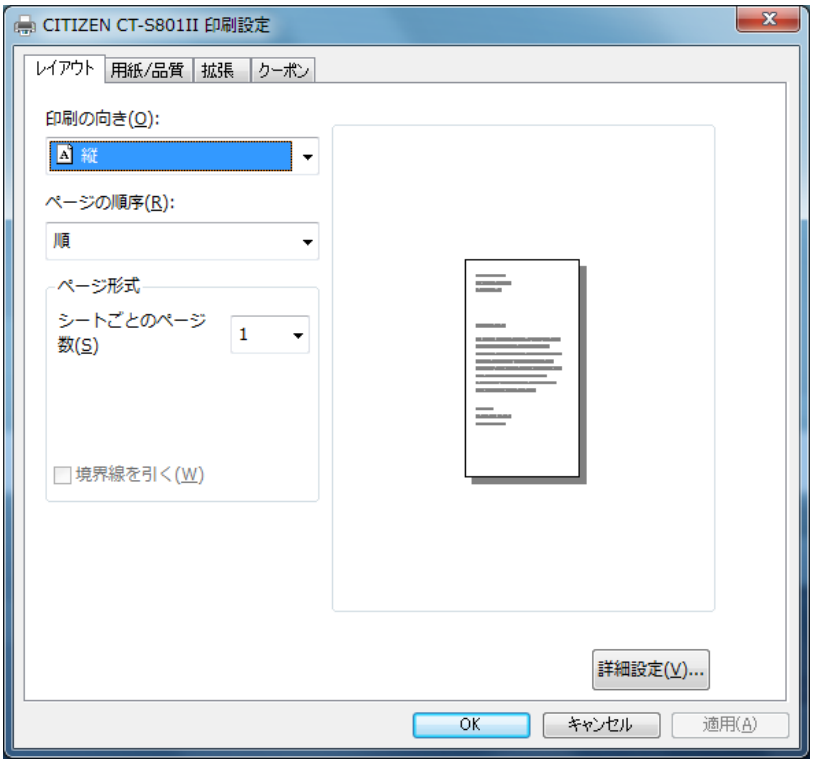


4. 3 電子ジャーナル

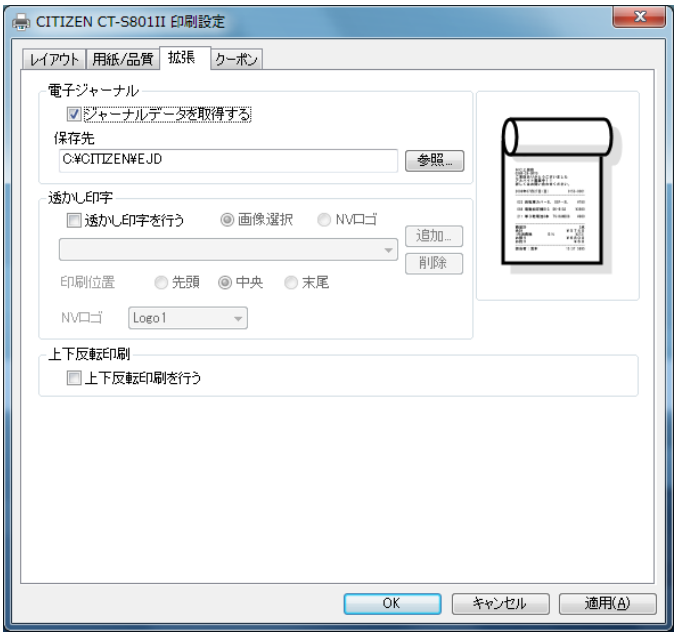
電子ジャーナルは印刷されたデータを保存する機能です。以下の画面より設定を行えます。

プリンタードライバーのプロパティ画面より、「基本設定」を選びます。

右のような画面が表示されますので、「拡張」タブを選びます。



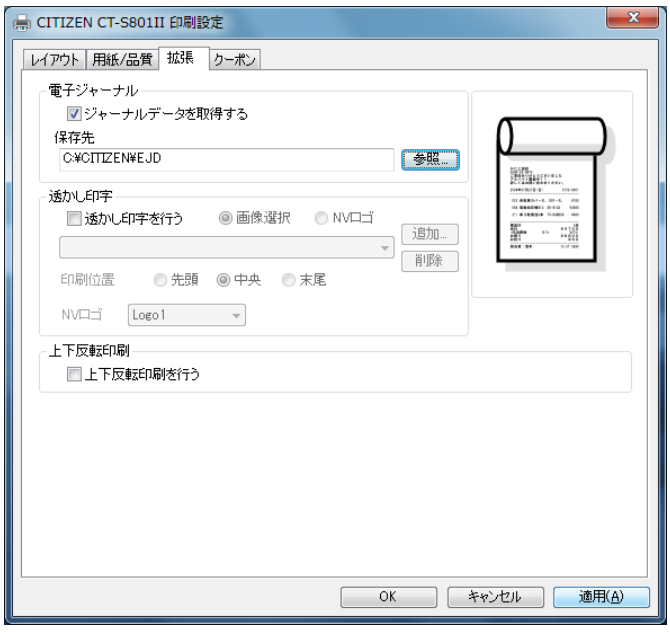
電子ジャーナルの機能を有効にするには「ジャーナルデータを取得する」にチェックを入れます。



保存先の「参照」を選ぶと、ジャーナルデータを保存するフォルダを指定できます。
デフォルトの格納先は「c:¥CITIZEN¥EJD」となっています。

「フォルダの参照」では新規フォルダの作成は出来ませんので、任意のフォルダを予め用意しておいてから、指定して下さい。

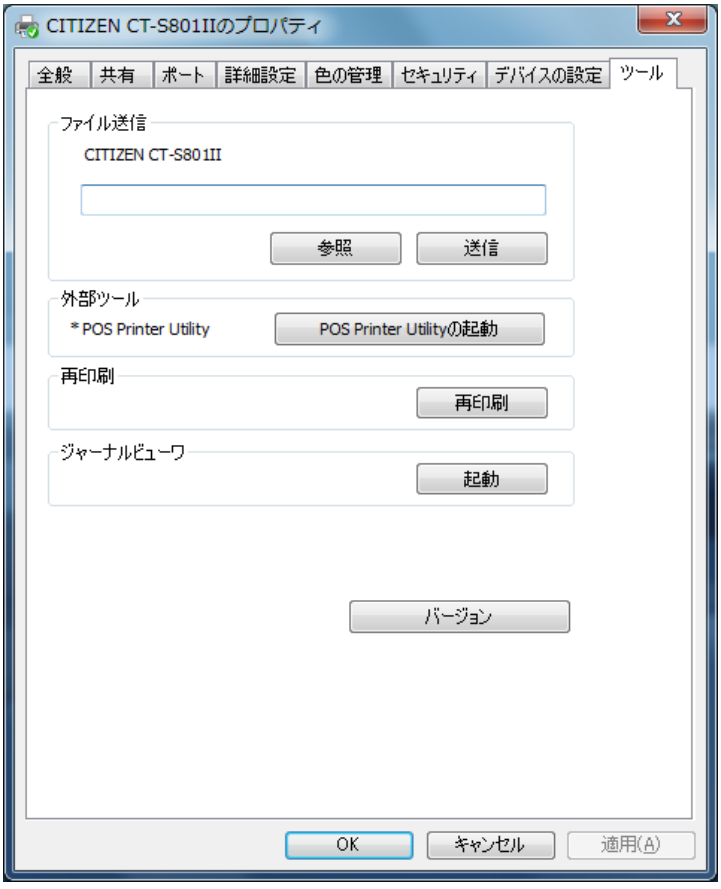
「適用」または「OK」を選んで、設定完了です。



電子ジャーナルの表示／印刷はジャーナルビューワより行う事ができます。

プリンタードライバーのプロパティ画面より、「ツール」タブを選びます。

ジャーナルビューワの「起動」を押します。



ジャーナルビューワが起動します。ここでジャーナルデータの表示／印刷を行う事が出来ます。

・ジャーナルデータのファイルは指定フォルダ（保存先の「参照」で指定したフォルダ）の直下に、日付毎のフォルダが作成され、その中に格納されます。フォルダ名は、年月日をとって yyyymmdd となります。ジャーナルデータのファイル名は、取得時分秒をとって、hhmmss.ejd となります。

・1 印刷ジョブデータ毎に 1 ジャーナルデータとして保存されていきます。

・ジャーナルデータの保存先のパス（文字列）長さは、100 文字（Unicode）以下として下さい。

・ネットワークドライブをジャーナルデータの保存先として、指定しないで下さい。

4. 3. 1 ジャーナルビューワ

・ジャーナルデータは年月日単位でフォルダ別けされています。「日付指定」のカレンダーより、指定した日付のジャーナルデータの表示／印刷を行う事が出来ます。ジャーナルデータが保存されている日には、その日付の数字が太字で強調されて表示されます。

・カレンダー上で右クリックすると、今日の日付に戻るメニューが表示されます。

・「参照先」はジャーナルデータが格納されているフォルダを指定します。通常、プリンタードライバーの電子ジャーナル保存先で指定したフォルダにします。他のフォルダを指定する場合、ジャーナルデータの参照先は 7 個の履歴を有していますので、この中から選択するか、新しく指定する場合は右側にある「参照」を押して指定して下さい。

・左側のビューワ部には、指定されたジャーナルデータのイメージが表示されます。その下にある「先頭データ」、「前データ」、「次データ」、「最終データ」を選ぶと、その日付内に含まれる各ジャーナルデータへ移動する事が出来ます。

・一番に下にある ejd ファイル指定欄より、ファイル名から直接参照する事も出来ます。

・「印刷」部にある「出力先:」には、原則としてジャーナルビューワを起動したプリンタードライバー名が表示されます。

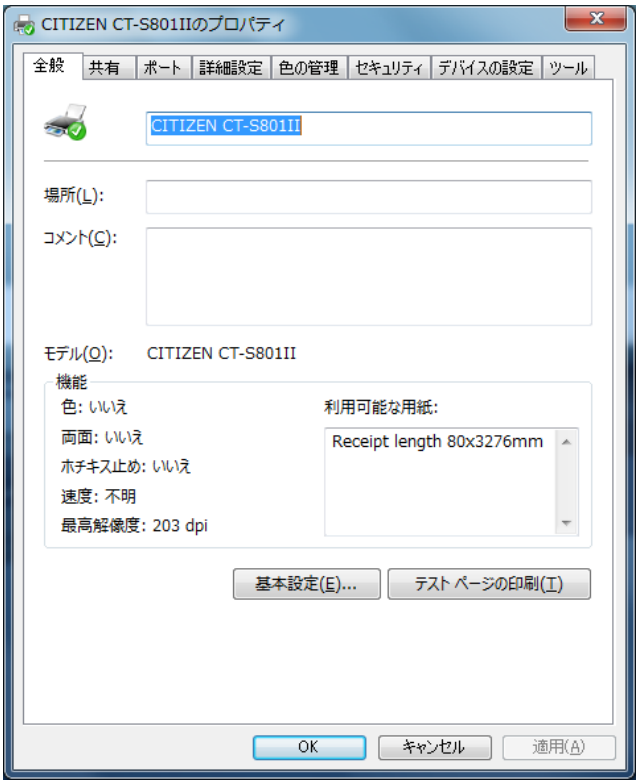
・「印刷実行」は「表示中の 1 件」または「指定日の 1 日分」のいずれかを選択して実行します。「指定日の 1 日分」を選択して「印刷実行」した場合は、各件の印刷の間にはカット動作が入らず、一気に印刷されます。

・プリンター本体側の個別機能である NV ロゴ印刷、プリンターフォント、バーコードフォントについてはビューワでは正しく表現されません。NV ロゴ印刷は NV ロゴの印刷を示す代替画像、バーコードフォントはバーコードシンボルの印刷を示す代替画像、二次元バーコードフォントは二次元コードの印刷を示す代替画像、プリンターフォントは一般的なフォントとして実際に印刷される内容とは違ったイメージで表示されます。
・特殊機能である Control フォントを含むジャーナルデータは、文字化けしているようなイメージが表示される事があります。
※プリンターフォント、バーコードフォント、二次元バーコードフォント、特殊機能については後述の章を参照下さい。
・ジャーナルデータ内に上下反転印刷機能は含まれません。つまりジャーナルビューワからの印刷は、プリンタードライバーの上下反転印刷設定内容によって印刷方向の影響を受けます。
・同様に用紙カット、ドロワー、ブザーの 3 つも、ジャーナルデータ作成時の設定は含まれません。ジャーナルビューワからの印刷時の設定の影響を受けます。
・「再印刷」されたデータはジャーナルデータとして扱われません。
・「ファイル送信」されたデータはジャーナルデータとして扱われません。
・ジャーナルビューワは、他機種プリンタードライバーで作成されたジャーナルデータも参照可能になっています。違う機種のプリンタードライバーで作成したジャーナルデータを参照して印刷しますと、間違った印刷を行ってしまう事がありますので、ご注意下さい。

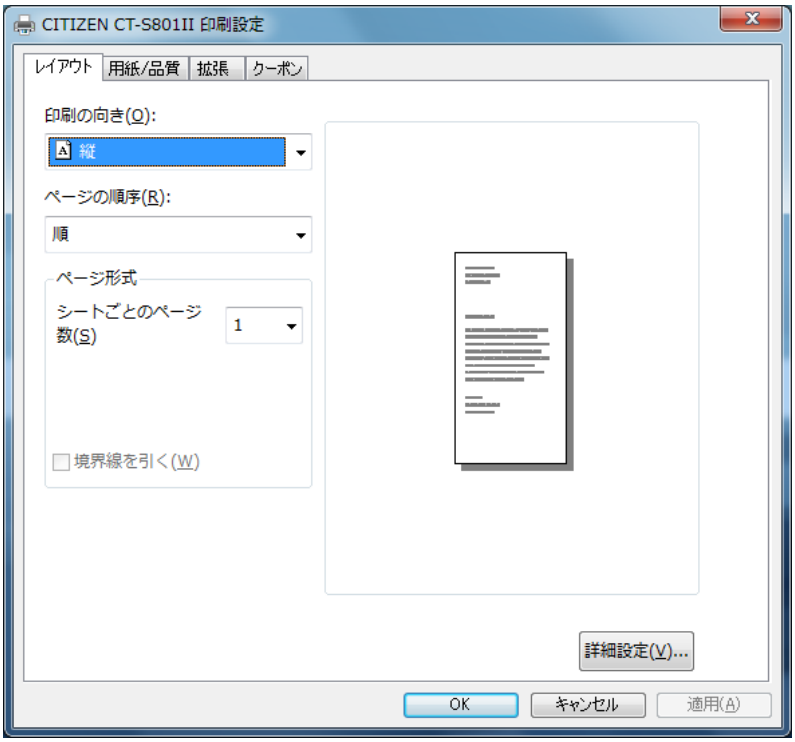
4. 4 透かし印字

印刷データに、予め用意しておいた画像イメージを重ねて印字させる事が出来ます。

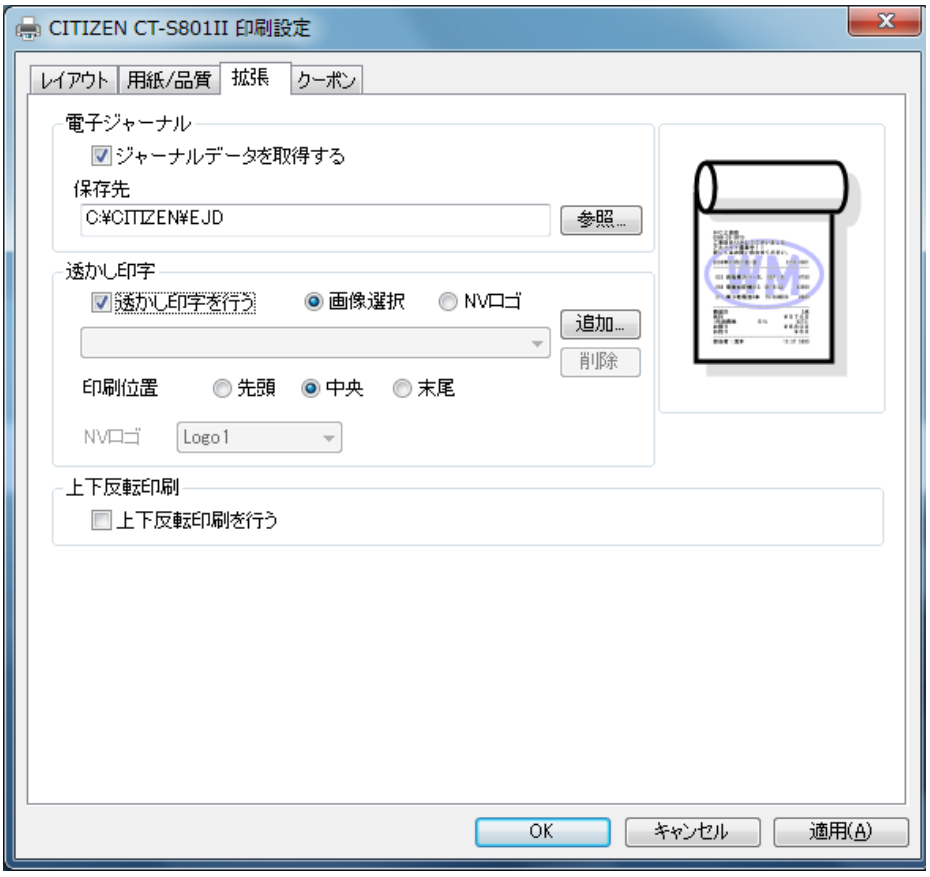
プリンタードライバーのプロパティ画面より、「基本設定」を選びます。



右のような画面が表示されますので、「拡張」タブを選びます。



透かし印字の機能を有効にするには「透かし印字を行う」にチェックを入れます。次に「画像選択」か「NV ロゴ」のどちらかを選択します。「画像選択」ではドライバー側で画像の重ね処理を行います。重ねる画像データを「追加」より指定し、重ねる位置を「先頭」、「中央」、「末尾」より選択します。「NV ロゴ」はプリンター側で画像の重ね処理を行います。プリンター本体に登録されている NV ロゴ(logo1～logo9)を指定します。
(NV ロゴでの透かし印字は、Ver3.00 以降だけの機能です)



・重ねる画像として登録出来るのはモノクロビットマップ形式だけとなります。Ver3.00 以降についてはグレースケール印刷用としてグレー(4bpp)のビットマップも登録出来ます。

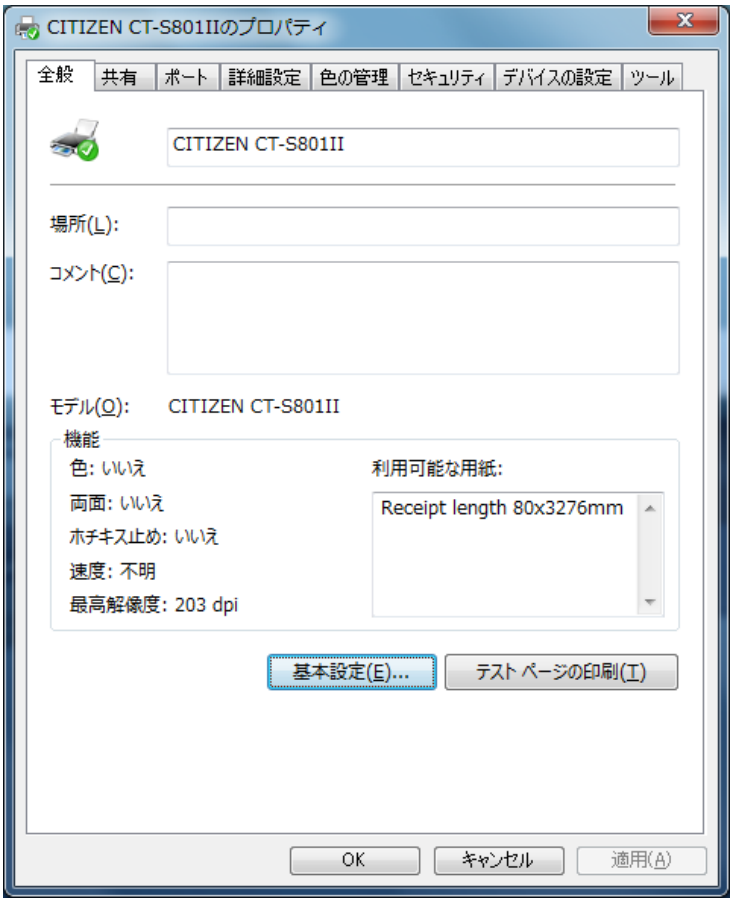
・横方向 1023 ピクセルまでのサイズの画像が登録可能です。縦方向の制限はありません。また実際の印刷時には用紙サイズの制限を受けます。
・予め 5 つまでの画像が登録可能です。それ以上登録させる場合は、「削除」にて既存の物を消してから登録して下さい。
・「画像選択」、「NV ロゴ」のどちらにおいても、より綺麗な透かし印字を行うにはグレースケール（階調）印刷をご利用下さい。
・「NV ロゴ」は重ねる位置の指定は無く、繰り返し重なって印刷されます。

・印刷データに画像イメージを重ね処理には以下の制限がありますので、予めご了承下さい。
縦 24 ドット基準で Windows から受け取ったイメージブロックと登録画像を重ねて処理しているため、
(1)受け取ったイメージブロックが、高さ 24 ピクセルより大きい行では、登録画像が縦方向の隙間を空けて切れて印刷される。特に大きなサイズのプリンターフォントを扱っている場合に顕著に見られる。
(2)受け取ったイメージブロックが、高さ 24 ピクセルより小さい行では、イメージブロック側を 24 ピクセルに広げてから、画像を重ねて出力する。よって結果的に間延びされて印刷されるようになる。
・バーコードフォントと画像イメージを重ねる事は出来ません

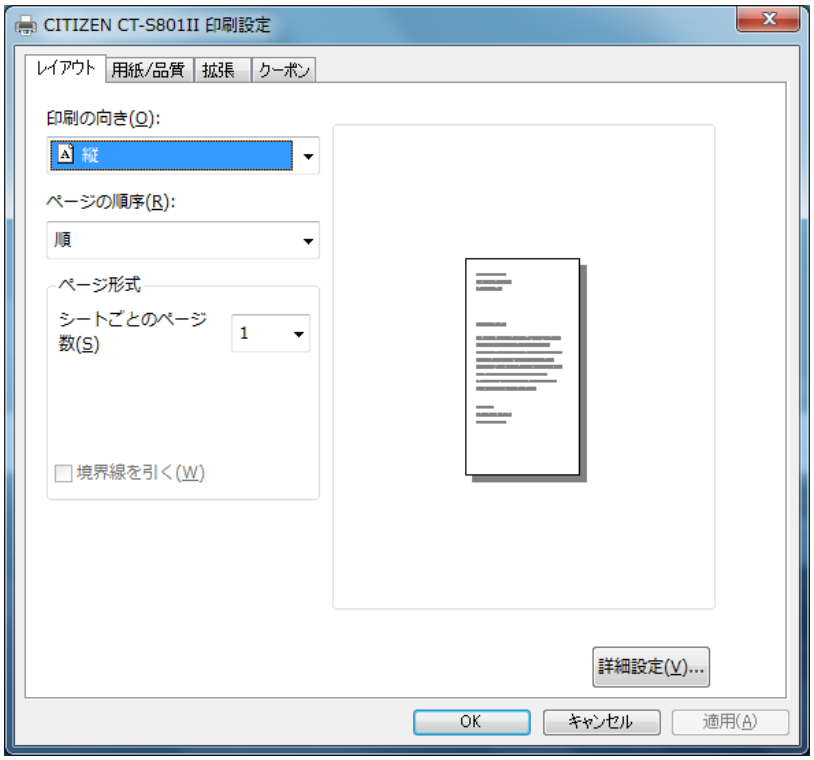
4. 5 上下反転印刷

印刷データを上下反転させて印刷させる事が出来ます。

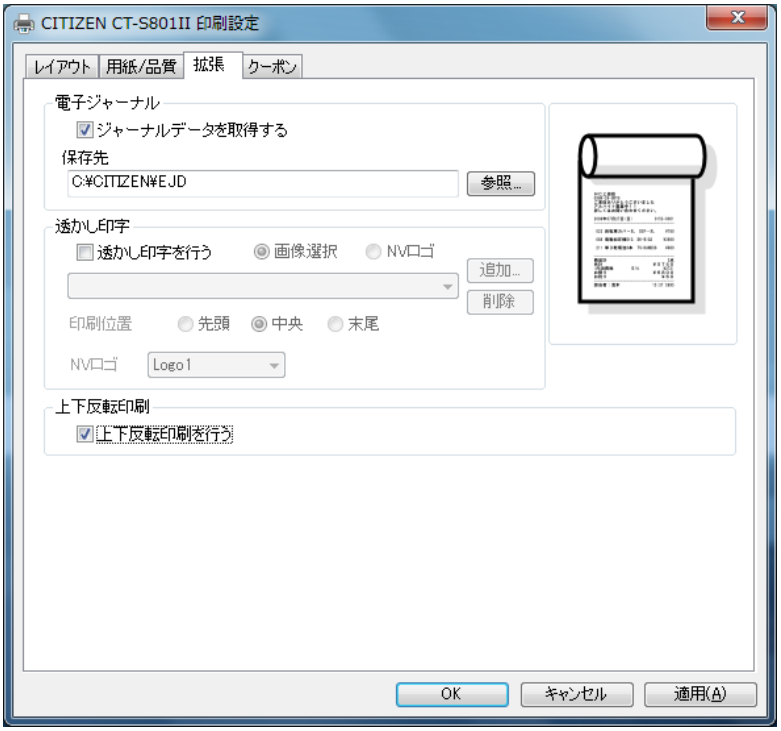
プリンタードライバーのプロパティ画面より、「基本設定」を選びます。



右のような画面が表示されますので、「拡張」タブを選びます。



上下反転印刷の機能を有効にするには、「上下反転印刷を行う」にチェックを入れます。



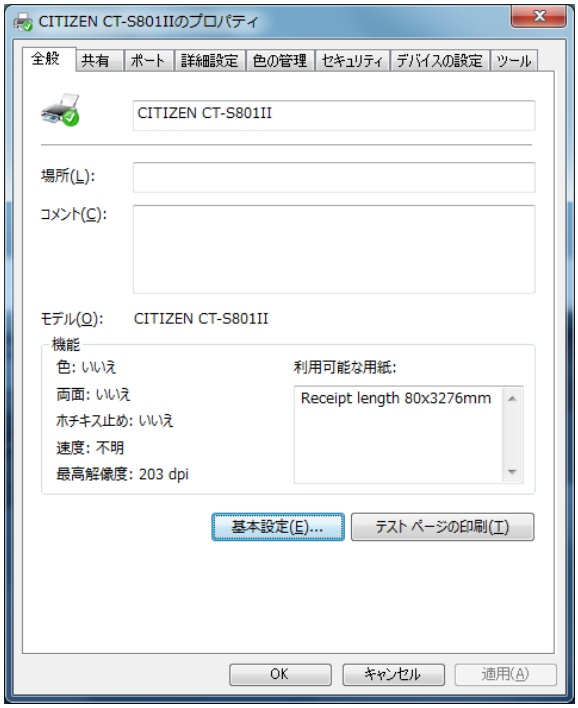
・プリンターフォントの上下反転も行えます。

・GS (L 方式で登録した NV ロゴは反転印刷出来ません。予め上下反転したイメージを登録してお使い下さい。

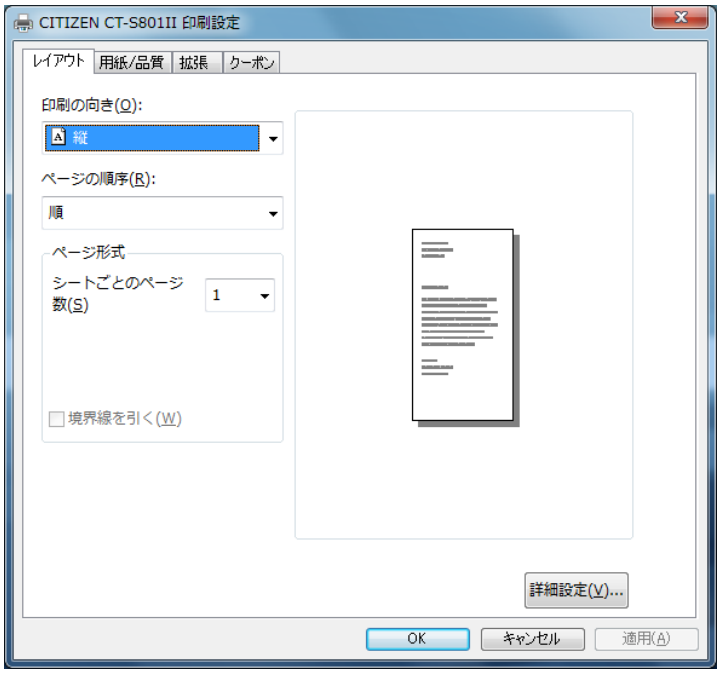
4．6 クーポン印刷

クーポン印刷とは、印刷データ内から特定の文字列キーワードを検索し、一致した場合、そのキーワードに関連付けられているクーポンを印刷するという機能です。

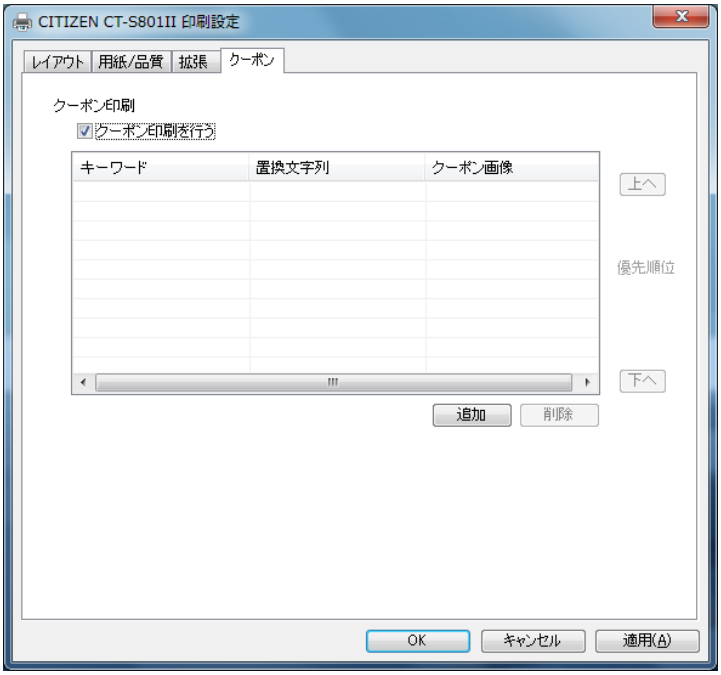
プリンタードライバーのプロパティ画面より、「基本設定」を選びます。



右のような画面が表示されますので、「クーポン」タブを選びます。



クーポンの機能を有効にするには「クーポン印刷を行う」にチェックを入れます。



クーポン情報の登録

・「追加」を押すと登録ダイアログが表示されます。クーポン情報は「キーワード」、「置換文字列」「クーポン画像」の 3 つをセットとして登録します。

・クーポン情報は 10 件まで登録出来、優先順位がつきます。クーポン情報欄の上～下は、優先順位の高～低を示しています。「上へ」、「下へ」で優先順位の変更が出来ます。
キーワード検索→文字列置換→クーポン画像出力は、優先順位の高いクーポン情報から行われます。

・登録出来るクーポン画像はモノクロビットマップ形式で、Ver3.00 以降はグレースケール印刷用としてグレー(4bpp)のビットマップも登録出来ます。横幅は 1023 ピクセルまでのデータを登録出来ます。

・"Keyword"、"Replacing Characters"、"Coupon Image"が同じ内容のクーポン情報は登録出来ません。いずれか一つが異なる場合は別のクーポン情報として扱われ、登録する事が可能です。

クーポン印刷の動き

・まず印刷データ先頭から「キーワード」と一致する文字列が存在するかどうかの検索作業が行われます。一致が検出されると、関連付けられている「クーポン画像」データを印刷データの最後に配置します。その後、印字データ内の一致された文字列に対し、「置換文字列」への置換えを行います。

・クーポン情報が複数ある場合、『クーポン画像出力』と『文字列置換』の 2 つの作業に大別されます。

・まず『クーポン画像出力』が先に行われます。優先順位に従って印刷データ先頭から「キーワード」検索が行われます。一致を検出した場合、関連付けられた「クーポン画像」が印刷データ最後に出力されます。そして次の優先順位のクーポン情報へ移り、同様の作業が行われます。最後の一番優先順位の低いクーポン情報の検索作業を終えると『クーポン画像出力』の終了です。

・次に『文字列置換』が行われます。優先順位に従って印刷データ先頭から「キーワード」検索が行われます。一致された文字列は関連付けされている「置換文字列」へ置換えられ、印刷データの最後までこの作業が行われます。そして次の優先順位のクーポン情報へ移り、印刷データ先頭から同様の処理が行われます。最後の一番優先順位の低いクーポン情報の検索作業を終えると『文字列置換』の終了です。

・以上のような処理の流れから、『文字列置換』に関しては、高い優先順位のクーポン情報により置換された文

字列が、それより低い優先順位のクーポン情報により、再置換させる事が出来ます。

※クーポンにおける制限事項

文字に関する制限

・TrueType フォント使用時は文字列置換が働きません。文字列置換を行う場合はプリンターフォントを使用して下さい。
・バーコードフォントは文字列置換出来ません。
・Control フォントは文字列置換出来ません。
・複数のフォントで構成されている、または異なるフォントサイズで構成されている文字列は、一つの文字列の塊として処理されないため、キーワードとして検索出来ません。
・複数行にまたがる文字列は、各行毎に別けて処理されてしまうため、キーワードとして検索出来ません。
・TrueType の記号フォント（Webdings、MS Extra 等）はキーワードとして検索出来ません。
・キーワードと検索文字列には以下の関係があります。
「キーワード」長＞「置換文字列」長 ：置換した結果、前に詰まる。
「キーワード」長＜「置換文字列」長 ：置換文字列をキーワードの長さ分まで置換。その後は切捨て。

アプリケーションに関する制限

・MS-Word では同一行で同フォント、同フォントサイズのデータであっても、2 つ以上に分割されて文字列が処理される事があります。この場合、分割された位置にある文字列はキーワードとして検索出来ません。
・Ms-Word では「"」や「'」は、オートコレクト機能で「”」や「’」に変換されてしまう事があります。この場合キーワード検索出来なくなります。
・Notepad はデバイスフォントを扱えないので、プリンターフォントを使用する事が出来ません。全て TrueType フォント扱いとなります。
・余白設定を 0 にすると、左右マージン位置にある文字は印字されないため、その文字を削除して扱われる事があります。この場合、この位置にあたる文字列はキーワードとして検索出来ない事がありますのでご注意下さい。
・VB や VC ではメソッドや API 上で、用紙幅を超えるサイズの文字数を送る事が出来てしまいます。しかしながら用紙幅を超えた分の文字は、当然無視されてしまいますので、この部分の文字列はキーワードとして検索は出来ません。用紙幅に収まる範囲でご使用下さい。

クーポン出力に関する制限

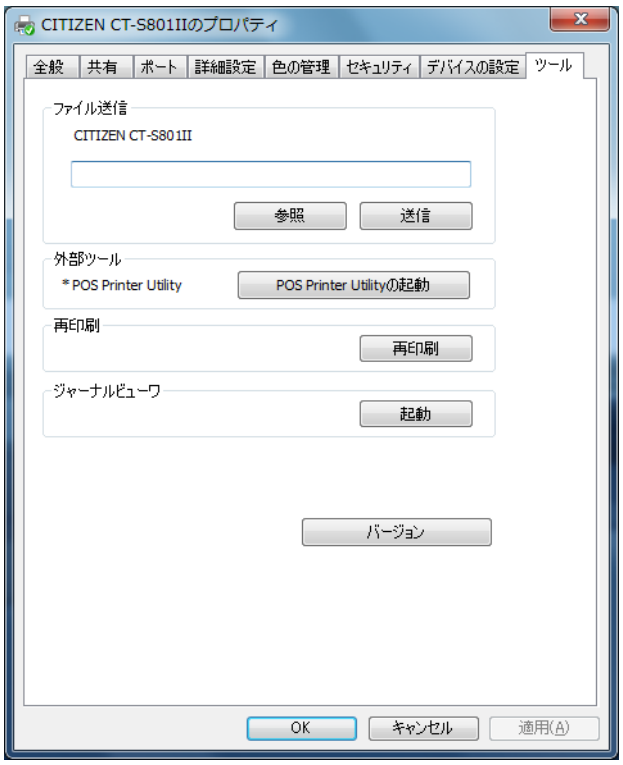
・文書最後に NV ロゴ印刷が指定されている場合、クーポンは NV ロゴの後に印刷されます。
・複数ページの印刷の場合、途中ページで検索出力されたクーポンは最終ページの最後にまとめて印刷されます。
・複数部数の印刷の場合、各部数毎にクーポンが印刷されます。
・Paper Type が「Ticket」に設定されている場合、設定された用紙長さを出力後にクーポンが印刷されます。
・印刷の向きを横に設定してもクーポンは横向きには出力されません。

4. 7 ファイル送信

予め用意しておいた任意のデータファイルを送信する事ができます。プリンタードライバー側設定によるデータの付加は一切無く、データファイルそのものだけがプリンターへ出力されます。

プリンタードライバーのプロパティ画面より、「ツール」タブを選びます。

「ファイル送信」より、「参照」を押して送信するファイルを指定し、「送信」でプリンターへ送ります。



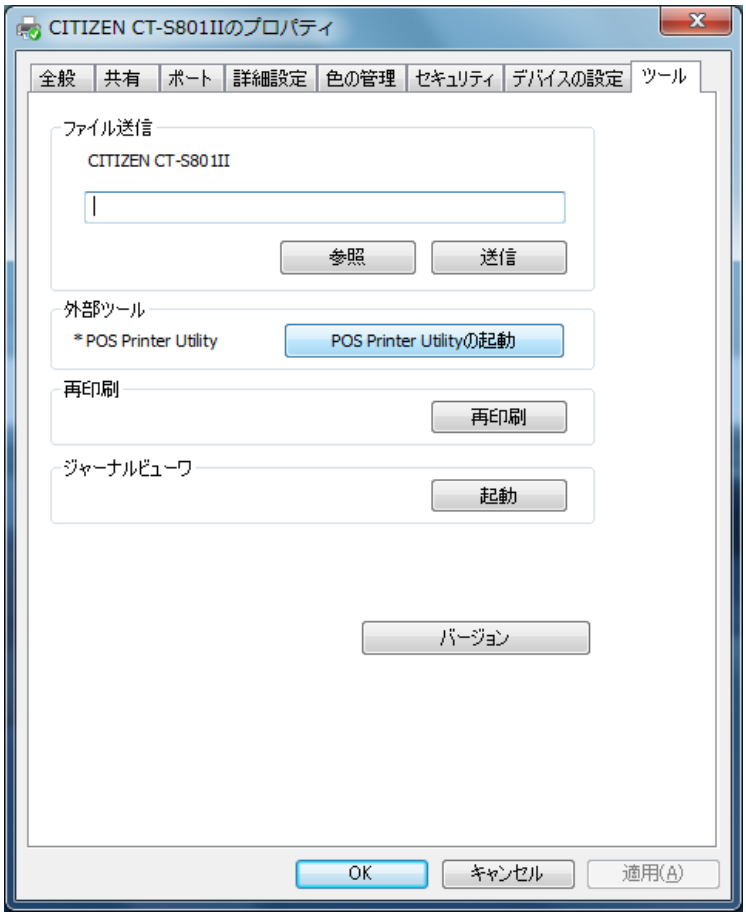
・ファームウェアデータの書換え作業等に利用出来ます。

4. 8 POS Printer Utility

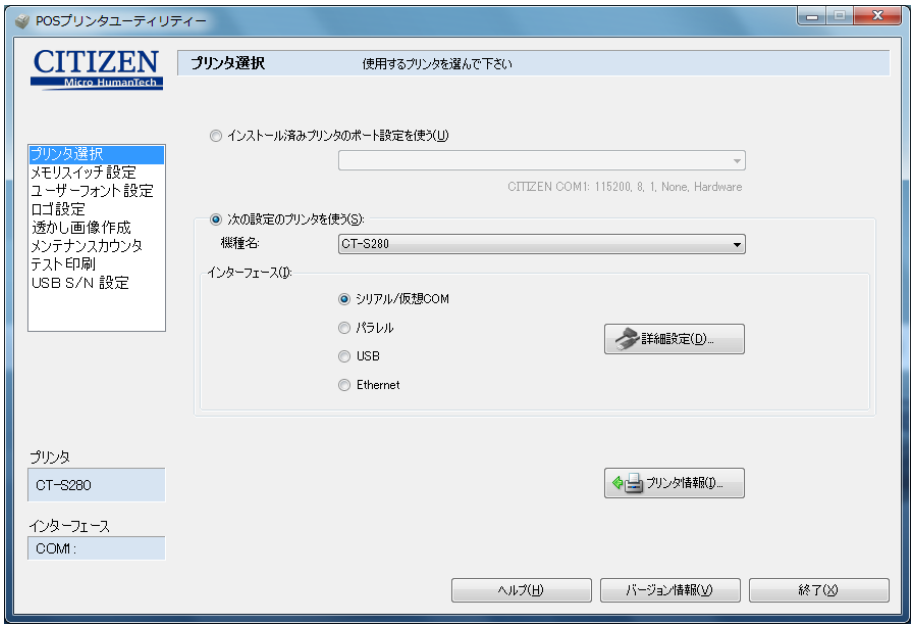
プリンターの様々な設定を行えるユーティリティを用意しています。以下の方法で起動出来ます。

プリンタードライバーのプロパティ画面より、「ツール」タブを選びます。

「外部ツール」より、「POS Printer Utility の起動」を押して下さい。



右のようなアプリケーション画面が表示されます。POS Printer Utility の使い方については、POS Printer Utility の HELP ファイルを参照下さい。

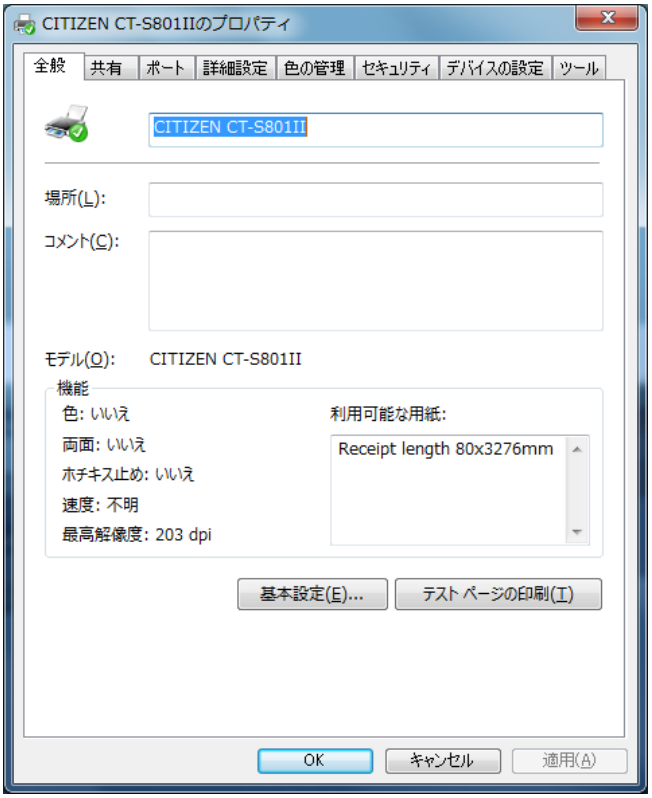


・ステータスモニター付ドライバーを使用している場合はプリンターとの通信に時間がかかります。
・ステータスモニター付ドライバーを USB、パラレル等の Plug & Play Port 経由でインストールした場合、インストール直後に POS Printer Utility を使用すると、通信に失敗する事があります。PC を再起動する事で安定して通信出来るようになります。

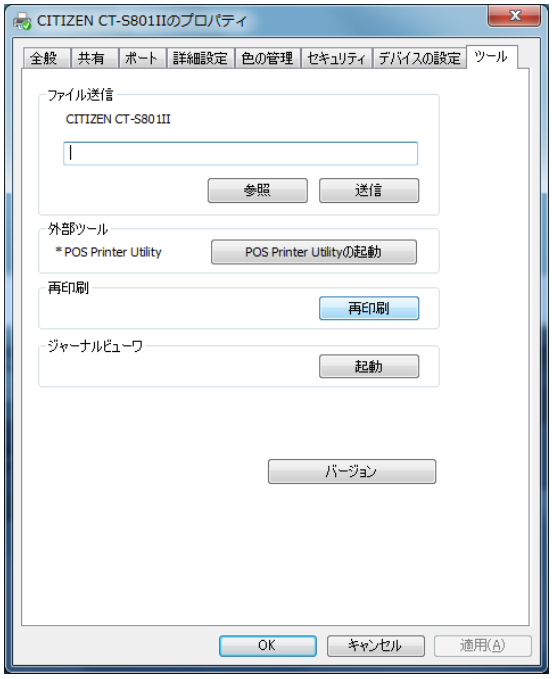
4. 9 再印刷

直前に印刷したデータを、再度印刷出来ます。

プリンタードライバーのプロパティ画面より、「ツール」タブを選びます。



「再印刷」より、「再印刷」を押す事で実行出来ます。



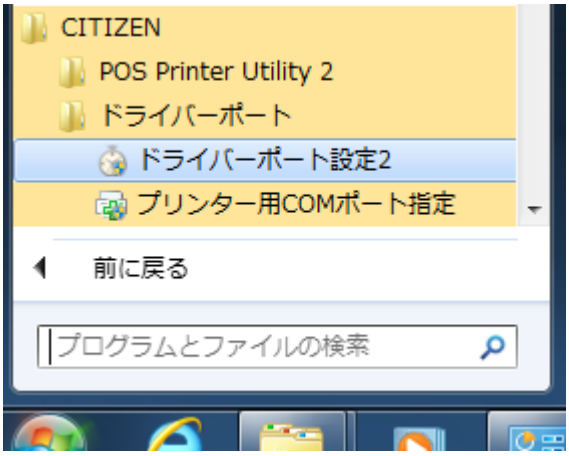
・PC を再起動しても、再起動前に印刷を行っていれば、そのデータの再印刷が行われます。
・再印刷は 1 印刷ジョブ単位です。すなわち直前の印刷が、複数ページにまたがる印刷データであった場合、最後のページだけが再印刷されるのではなく、直線の印刷ジョブに含まれる全ページが再印刷されます。
・上下反転印刷は再印刷実行時の設定の影響を受けます。
・再印刷のデータは電子ジャーナルデータとして扱われません。
・「ファイル送信」で送信したデータファイルは、再印刷されません。

再印刷機能の実現にはジャーナルビューワがインストールされている必要があります。

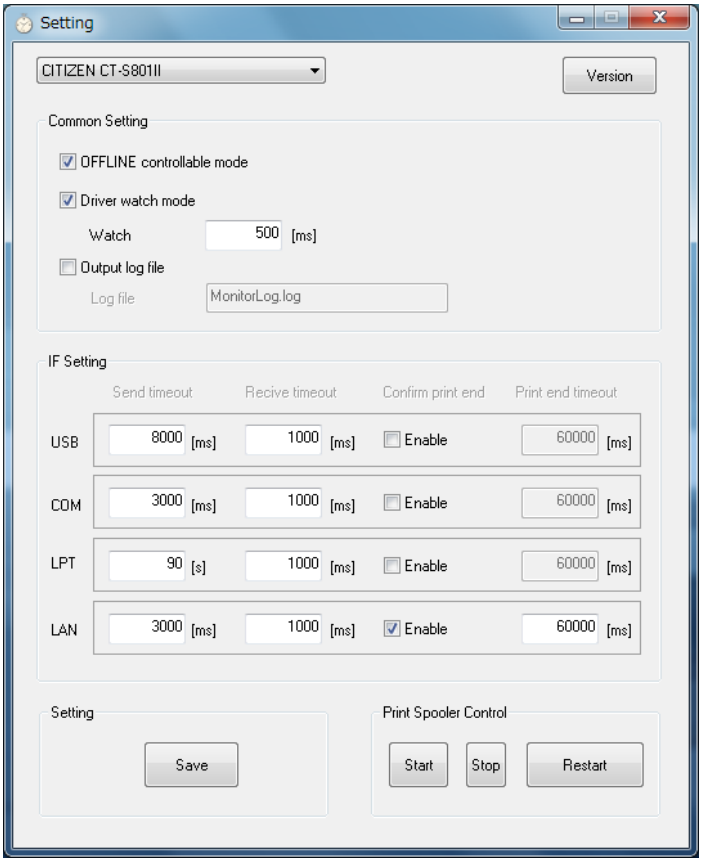
4. 10 ドライバーポート設定

ステータスモニター付ドライバー使用時に、各インターフェースポートのタイムアウト時間等を設定します。

スタートメニューより、「CITIZEN」－「ドライバーポート」－「ドライバーポート設定2」を指定します。



右のようなダイアログが起動します。



・USB、COM、LPT、LAN の各送信タイムアウト、受信タイムアウト時間を設定する事が出来ます。

・設定方法は、各インターフェースのタイムアウト時間を任意に変更し、「Save」ボタンを押します。その後、プリントスプーラーの「Restart」ボタン（または「Stop」ボタン→「Start」ボタン）を押して下さい。設定が反映されます。

・「Driver watch mode」にチェックが入っている場合は常にプリンターステータスを監視している状態となります。「Watch xxx [ms]」が監視時間間隔となります。チェックが外れている場合はユーザーのプログラムにより、プリンターステータス取得 API が呼ばれた時だけ、プリンターステータスの監視を行います。設定変更は、タイムアウト設定と同じように「Save」ボタン押した後に、プリントスプーラーをリセットして下さい。通常は「Driver watch mode」にチェックが入っている状態でご使用下さい。

・「OFFLINE controllable mode」はプリンターエラーが発生した時、印刷スプーラーをオフラインにする／しないを選択できる機能です。チェックを外すとオフラインにならなくなります。

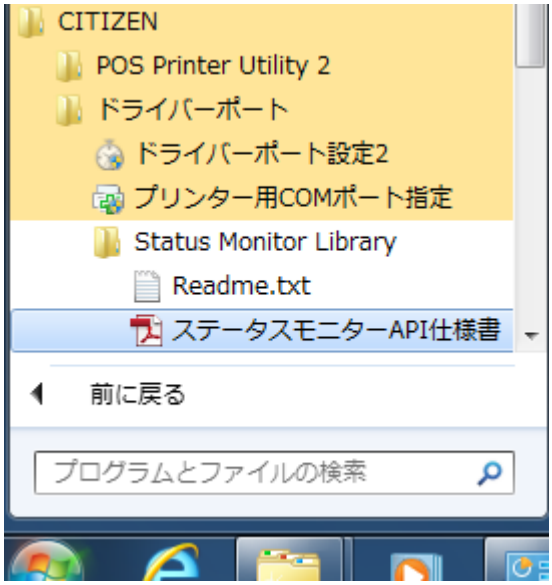
・各インターフェースにはデフォルト値として、最適な値が設定されています。通常変更する必要はありません。しかし、LPT の送信タイムアウト時間だけは、OS の初期値として 90 秒になっています。90 秒という時間は長すぎますので、LPT をご使用になる場合は推奨値の 5 秒に設定して下さい。

・環境によりプリンターステータスが取得し難い場合、送信タイムアウト値を多めに設定して下さい。通常これで取得出来るようになります。

4. 11 ステータスモニターライブラリー

ステータスモニター機能は、CITIZEN 独自のステータス取得方法を提供しています。ステータスモニター API を駆使してプログラミングする事により、OS が提供しているプリンター情報より多くのステータスを得る事が出来ます。詳細は「ステータスモニターAPI 仕様書」と「Readme.txt」をご覧下さい。「Readme.txt」にはサンプルソースコードの格納場所が記載されています。

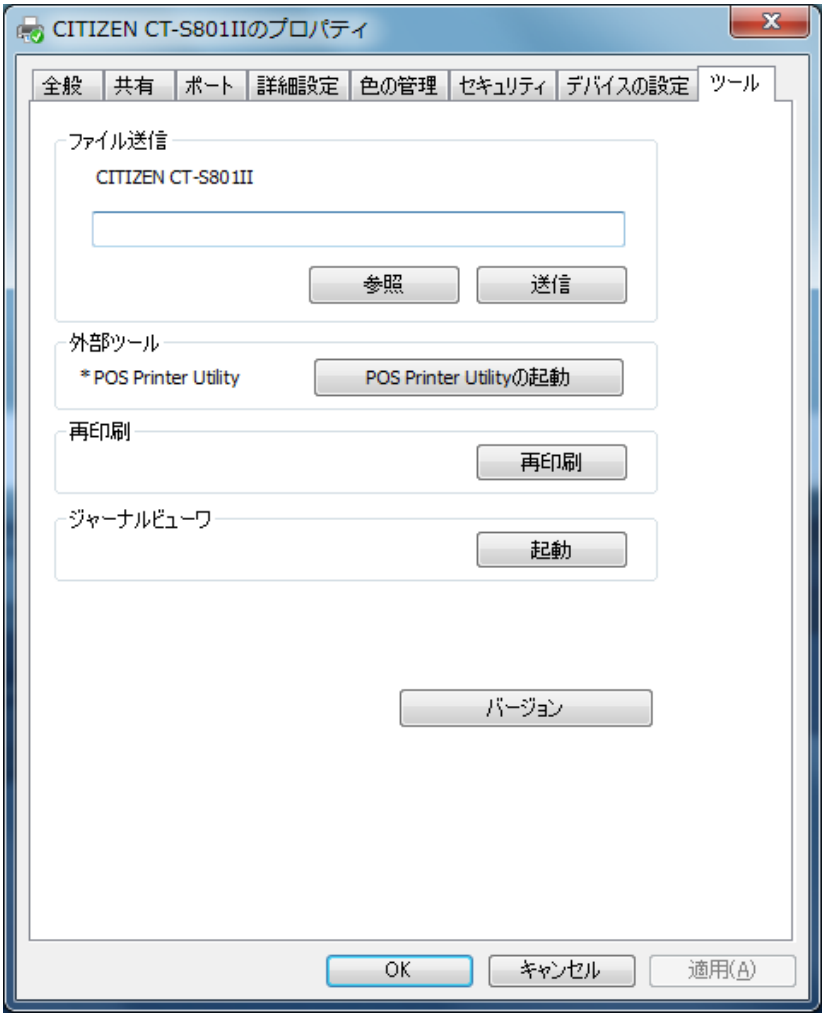
スタートメニューより、「CITIZEN」－「ドライバーポート」－「Status Monitor Library」にあります。



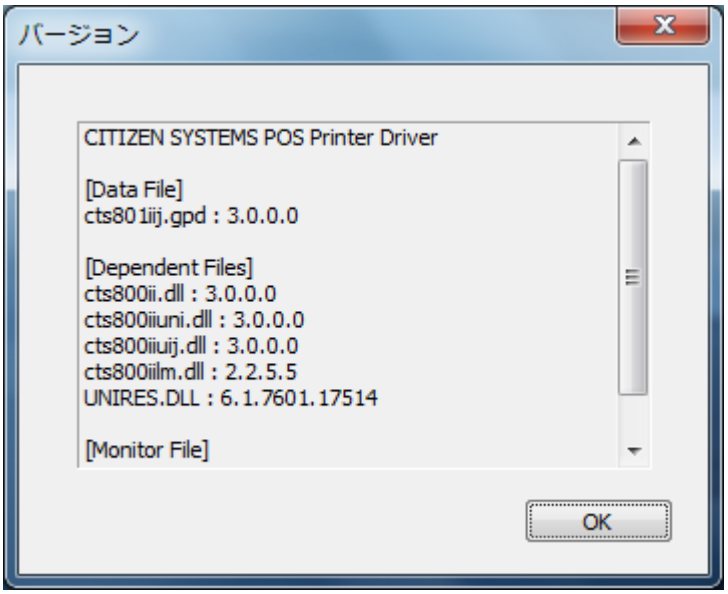
4. 12 バージョン表示

プリンタードライバーを構成するファイルのバージョン一覧を表示します。

プリンタードライバーのプロパティ画面の「ツール」タブより「バージョン」ボタンを押します。



ドライバーを構成するファイルのバージョン一覧が表示されます。



5. プリンターフォント

プリンターフォントはプリンターに内蔵されている文字の事を言います。プリンターフォントによる印刷は TrueType フォントによる印刷と比べて、非常に早い印刷を行えます。早い印刷のお望みの場合は、プリンターフォントを使う事をお勧めします。

プリンタードライバーは以下のプリンターフォントをサポートしています。

ドライバー	フォント		フォントサイズ
CBM1000II CT-S280 CT-S281 CT-S300 CT-S310 CT-S310II CT-S601(II) CT-S651(II) CT-S801(II) CT-S851(II) CT-S2000 CT-S4000 CT-P29x PMU2xxx PMU2xxxIIIPresenter PPU-700 BD2-222x BD2-428x	15 cpi 15 cpi (RED) 7.5 cpi 7.5 cpi (RED) 3.75 cpi 3.75 cpi (RED) 1.8 cpi 1.8 cpi (RED)	フォント A	12 / 24 / 36 / 48 / 60 / 72 / 80
	20 cpi 20 cpi (RED) 10 cpi 10 cpi (RED) 5 cpi 5 cpi (RED) 2.5 cpi 2.5 cpi (RED)	フォント B	
	25 cpi 25 cpi (RED) 12.5 cpi 12.5 cpi (RED) 6.25 cpi 6.25 cpi (RED) 3 cpi 3 cpi (RED)	フォント C	
CBM1000II CT-S280 CT-S281 CT-S300 CT-S310 CT-S310II CT-S601 CT-S651 CT-S801 CT-S851 CT-S2000 CT-S4000 CT-P29x PMU2xxx PMU2xxxIIIPresenter PPU-700 BD2-222x BD2-428x	FontA11 FontA11 [255]	フォント A	9.5
	FontA12 FontA12 [255]		19
	FontA21 FontA21 [255]		9.5
	FontA22 FontA22 [255]		19
	FontA24 FontA24 [255]		38.5
	FontA42 FontA42 [255]		19
	FontA44 FontA44 [255]		38.5
	FontA48 FontA48 [255]		77
	FontA84 FontA84 [255]		38.5
	FontA88 FontA88 [255]		77
	FontB11 FontB11 [255]	フォント B	7 (CT-S3x0/6x1(II)/8x1(II)) 9.5 (上記以外のモデル)
	FontB12 FontB12 [255]		13.5 (CT-S3x0/6x1(II)/8x1(II)) 19 (上記以外のモデル)

	FontB21 FontB21 [255]		7 (CT-S3x0/6x1(II)/8x1(II)) 9.5 (上記以外のモデル)
	FontB22 FontB22 [255]		13.5 (CT-S3x0/6x1(II)/8x1(II)) 19 (上記以外のモデル)
	FontB24 FontB24 [255]		27 (CT-S3x0/6x1(II)/8x1(II)) 38.5 (上記以外のモデル)
	FontB42 FontB42 [255]		13.5 (CT-S3x0/6x1(II)/8x1(II)) 19 (上記以外のモデル)
	FontB44 FontB44 [255]		27 (CT-S3x0/6x1(II)/8x1(II)) 38.5 (上記以外のモデル)
	FontB48 FontB48 [255]		54.5 (CT-S3x0/6x1(II)/8x1(II)) 77 (上記以外のモデル)
	FontB84 FontB84 [255]		27 (CT-S3x0/6x1(II)/8x1(II)) 38.5 (上記以外のモデル)
	FontB88 FontB88 [255]		54.5 (CT-S3x0/6x1(II)/8x1(II)) 77 (上記以外のモデル)
	漢字 7.5cpi 漢字 7.5cpi (RED) 漢字 3.8cpi 漢字 3.8cpi (RED)	日本語フォント A	12 / 24
	漢字 10cpi 漢字 10cpi (RED) 漢字 5cpi 漢字 5cpi (RED)	日本語フォント B (CT-S2000)	
	漢字 13cpi 漢字 13cpi (RED) 漢字 6.5cpi 漢字 6.5cpi (RED)	日本フォント C	
	Control		12

CD-S500 シリーズ	16 cpi 16 cpi（RED) 8 cpi 8 cpi（RED)	9x9ドット	12 / 24 / 36 / 48 / 60 / 72 / 80
	12 cpi 12 cpi（RED) 6 cpi 6 cpi（RED)	7x9ドット	
	漢字 13cpi 漢字 13cpi（RED) 漢字 6.5cpi 漢字 6.5cpi（RED)	日本語フォント	12 / 24
	FontA11 FontA11 [255]	9x9ドット	9
	FontA12 FontA12 [255]		18
	FontA21 FontA21 [255]		9
	FontA22 FontA22 [255]		18
	FontB11 FontB11 [255]	7x9ドット	9
	FontB12 FontB12 [255]		18
	FontB21 FontB21 [255]		9
	FontB22 FontB22 [255]		18
	Control		12

・「xx cpi」、「xx cpi (RED)」は、xx 部分の数字が小さいほど、文字が横方向に大きくなります。標準サイズに対し、横 2 倍、4 倍、または 8 倍に拡大した物となります。
・「xx cpi」、「xx cpi (RED)」は、フォントサイズ 12 がこのフォントの標準となります。数字が大きくなると縦方向に拡大されていきます。
・「FontAxx」、「FontBxx」というフォントは、xx の部分が縦横の拡大比率を示しています。標準サイズは 11、縦 4 倍、横 8 倍なら 48 となっています。
・「xx cpi」と「FontAxx (FontBxx)」では、フォントサイズが違うことにより、改行量が異なります。
・サポートしている文字コード表は、WPC1252（日本語フォントでは、JIS）です。但し、[255]と付いているフォントでは、ユーザー登録フォント（Shift-out 側：80h～FFh 内に定義されるユーザー文字）を使用します。
・(RED)と付いているフォントを使うと 2 色（通常は赤と黒）印字が出来ます。但しプリンターの機種により、2 色印字を行うために必要な条件（2 色印字モードの設定有無、2 色印字サーマル用紙使用の有無など）が違いますので、お問合せ下さい。
・プリンタードライバーの設定の「印刷の向き」で、横方向印刷が選択されている時は、プリンターフォントの印刷は出来ません。
・「Control」フォントは、特殊機能（保証外となります）のみに使われ、印字はされません。詳細は、後述の章「特殊機能」をご覧下さい。
・アプリケーションが持っている、中央位置揃えや右揃えの機能は、プリンターフォントでは正しく動きません。
・プリンターフォントで一行の桁数を越えるデータを送ると、超えた部分のデータは原則として無視される仕組みになっています。しかしながら、送る手段によっては最後の一文字が次行に印字される事もあります。文字数が一行の桁数を超えないように調整下さい。
・プリンターフォント名には半角スペースが入っている物がありますので、ご注意下さい

␣は半角スペース

「15␣cpi」

「15␣ cpi␣(RED)」

「FontA11␣ [255]」

「漢字␣7.5cpi」

「漢字␣7.5cpi␣ (RED)」

・グレースケール（階調）印刷時はプリンターフォントの印刷は出来ません。

6. バーコード印刷

本ドライバーを使うサーマルプリンターでは、以下に示すフォントを使うとバーコードが印字出来ます。

バーコードフォント	フォントサイズ
Codabar	20 / 40 / 42 / 60 / 64 / 80 / 84
Code128	20 / 40 / 42 / 60 / 64 / 80 / 84
Code39	20 / 40 / 42 / 60 / 64 / 80 / 84
Code93	20 / 40 / 42 / 60 / 64 / 80 / 84
JAN13 (EAN)	20 / 40 / 42 / 60 / 64 / 80 / 84
JAN8 (EAN)	20 / 40 / 42 / 60 / 64 / 80 / 84
ITF	20 / 40 / 42 / 60 / 64 / 80 / 84
UPC-A	20 / 40 / 42 / 60 / 64 / 80 / 84
UPC-E	20 / 40 / 42 / 60 / 64 / 80 / 84

・プリンターに内蔵されているバーコード印刷機能を利用して実現しています。
・フォントサイズを選ぶ事で、異なるサイズのバーコードが印字出来ます。
・前述の章「Bar Code Printing」の設定により、上、下または上下に可視コードを印字する事が出来ます。
・印字幅を超えるサイズのバーコードは、印字されません。
・各バーコードには使える文字種の制限があります。詳細は製品に添付されている「コマンドリファレンス」の「バーコードコマンド」の章を参照して下さい。
・「Code128」のコードセットA、B、Cは、バーコード印字データの前に{A、{B、{Cのを付加して指定して下さい。
・「Code128」のコードセットCに入力するデータは、その文字の文字コードを、10進数に変換されたデータとして扱われ、印字されます。例えば、入力データ1 → 出力データ49（数字の1の文字コードは31h=49dec）。
・同一行にプリンターフォントとバーコードフォントの印字は出来ません。
・アプリケーションが持っている、中央位置揃えや右揃えの機能は、バーコードフォントでは正しく動きません。
・プリンター内蔵のバーコード印刷機能を利用しているバーコードフォントであるため、Windowsアプリケーションの画面上ではもちろんバーコードのイメージは表示されません。バーコード印字データである文字が表示されるだけとなります。
・プリンタードライバーの設定の「印刷の向き」で、横方向印刷が選択されている時は、バーコードフォントの印刷は出来ません。
・「JAN13 (EAN)」、「JAN8 (EAN)」のフォント名は、カッコの前に半角スペースがあります。
␣は半角スペース
「JAN13␣(EAN)」
「JAN8␣ (EAN)」
・グレースケール（階調）印刷時はバーコードフォントの印刷は出来ません。

7. 二次元バーコード印刷

7. 1 二次元バーコードフォント書式

本ドライバーがサポートする以下のプリンターフォントを用いて、二次元バーコードを印字出来ます。QR コードと PDF417 コードを印字出来ます。

二次元バーコードフォント	フォントサイズ	機能	命令文字	詳細
QR_CONTROL	9.5	QR コードコマンド設定	_M1	QR モデルの指定。モデル 1。
			_M2	QR モデルの指定。モデル 2。
			_W01 ～ _W16	QR モジュール幅の指定。1～16 ドット。
			_EL	エラー訂正レベル L。復元能力 7%。
			_EM	エラー訂正レベル M。復元能力 15%。
			_EQ	エラー訂正レベル Q。復元能力 25%。
			_EH	エラー訂正レベル H。復元能力 30%。
			_N****	QR データ数。4 桁の数字。0000～7089。
			_P	QR データの印刷。
QR_DATA	9.5	QR コードデータ設定	無	コード化したい QR データを指定します。
PDF417_CONTROL	9.5	PDF417 コードコマンド設定	_D00 ～ _D30	PDF417 の桁数指定。00 は自動処理。それ以外は指定桁数に指定する。
			_S00 _S03 ～ _S90	PDF417 の段数指定。00 は自動処理。それ以外は指定段数に指定する。
			_W2 ～ _W8	PDF417 のモジュール幅指定。
			_H2 ～ _H8	PDF417 の段の高さ指定。
			_e0 ～ _e8	エラー訂正レベル 0 ～ 8。
			_E01 ～ _E40	計算によるエラー訂正レベルの指定。
			_T0 _T1	簡易 PDF417 の解除。 簡易 PDF417 の指定。
			_N*****	PDF417 データ数。5 桁の数字。00000～65532。
			_P	PDF417 データの印刷。
PDF417_DATA	9.5	PDF417 コードデータ設定	無	コード化したい PDF417 データを指定します。

・プリンター本体が漢字コードをサポートしている場合、QR コード、PDF417 コードは漢字も扱う事が出来ます。
但しこの場合、漢字一文字→2 バイトとして、それぞれ QR データ数、PDF417 データ数を指定して下さい。

・二次元コードの指定可能なデータ数は書式上、QR コードでは 7089 バイト。PDF417 コードでは 65532 バイトとなっています。しかしながらシンボル化された二次元バーコードのサイズが、プリンターの印字領域サイズを越える場合は印刷されませんので、ご注意下さい。
・アプリケーションがサポートしている、中央位置揃えや右揃えの機能は、二次元バーコードプリンターフォントでは正しく動きません。
・プリンター内蔵のバーコード印刷機能を利用している二次元バーコード印刷であるため、Windows アプリケーションの画面上ではもちろん印刷プレビューにおいても、二次元バーコードのイメージは表示されません。二次元バーコード印字データである文字が表示されるだけとなります。
・プリンタードライバーの設定の「印刷の向き」で、横方向印刷が選択されている時は、二次元バーコードの印刷は出来ません。
・Visual Basic.Net, Visual C.Net ではプリンターフォントを使用出来ません。
・グレースケール（階調）印刷時は二次元バーコードの印刷は出来ません。

7．2 二次元バーコードフォントの使用例

二次元バーコードの印刷は、以下のような手順で行って下さい。
a) 二次元バーコード書式設定（QR_CONTROL、PDF417_CONTROL フォントを使用）
b) 二次元バーコードデータ設定（QR_DATA、PDF417_DATA フォントを使用）
c) 二次元バーコード印刷指令（QR_CONTROL、PDF417_CONTROL フォントで_P にて印刷）
※二次元コードデータ数の指定(QR_CONTROL の_N****, PDF417_CONTROL の_N*****)は、必ず、二次元コードデータ設定(QR_DATA, PDF417_DATA)の前で行って下さい。

二次元バーコードフォントを使用した VisualBasic のサンプルプログラムを示します。

```
'QR コード書式設定を行います
'モデル 2、モジュール幅 2、エラーレベル M、データ数 33
Printer.FontSize = 9.5
Printer.FontName = "QR_CONTROL"
Printer.Print "_M2_W02_EM_N0033";

'QR コードデータを指定します
Printer.FontSize = 9.5
Printer.FontName = "QR_DATA"
Printer.Print "http://www.citizen-systems.co.jp/";

'QR コードの印刷を行います
Printer.FontSize = 9.5
Printer.FontName = "QR_CONTROL"
Printer.Print "_P";

Printer.EndDoc
```

```
‘QR コード書式設定を行います
'モデル 2、モジュール幅 4、エラーレベル H、データ数 28(漢字データ)
Printer.FontSize = 9.5
Printer.FontName = "QR_CONTROL"
Printer.Print "_M2_W04_EH_N0028";

'QR コードデータを指定します
Printer.FontSize = 9.5
Printer.FontName = "QR_DATA"
Printer.Print "シチズン・システムズ株式会社";

'QR コードの印刷を行います
Printer.FontSize = 9.5
Printer.FontName = "QR_CONTROL"
Printer.Print "_P";

Printer.EndDoc



‘PDF417 コード書式設定を行います
'桁数 3、段数自動、モジュール幅 3、段の高さ 3、エラーレベル 01、データ数 22
Printer.FontSize = 9.5
Printer.FontName = "PDF417_CONTROL"
Printer.Print "_D03_S00_W3_H3_E01_N00022";

'PDF417 コードデータを指定します
Printer.FontSize = 9.5
Printer.FontName = "PDF417_DATA"
Printer.Print "PDF417-Code Test Print";

'PDF417 コードの印刷を行います
Printer.FontSize = 9.5
Printer.FontName = "PDF417_CONTROL"
Printer.Print "_P"


Printer.EndDoc
```

8. グラフィック印刷

プリンターの解像度は以下の通りです。

		解像度(DPI)
CD-S500 シリーズ	High quality	144 x 144
	Standard	144 x 72
	Draft	72 x 72
サーマルプリンター		203 × 203

TrueType フォントの印刷は、グラフィックにて印字されます。

・印刷の向きを横方向にして大きなサイズのグラフィック印字をすると、途中で 1mm 程度の隙間が開く場合があります。
・グラフィック印字は印刷データサイズが大きくなります。そのためシリアルインターフェースは、他のインターフェースと比べて通信スピードが遅いため、グラフィック印字には適していません。

9. 特殊機能

特殊機能を使う事でプリンターのいろいろな機能を使う事が出来ます。下記表の文字を「Control」フォントを指定してプリンターへ印字する事で利用出来ます。これ「Control」フォント以外が利用した場合、これらの機能は動作せず、印字データとして扱われます。

モデル	機能	"Control"フォント	
		ASCII	16 進数
ドロワーモデル	50 ミリ秒間 1 番のドロワーを駆動	A	41
ドロワーモデル	100 ミリ秒間 1 番のドロワーを駆動	B	42
ドロワーモデル	150 ミリ秒間 1 番のドロワーを駆動	C	43
ドロワーモデル	200 ミリ秒間 1 番のドロワーを駆動	D	44
ドロワーモデル	250 ミリ秒間 1 番のドロワーを駆動	E	45
ドロワーモデル	50 ミリ秒間 2 番のドロワーを駆動	a	61
ドロワーモデル	100 ミリ秒間 2 番のドロワーを駆動	b	62
ドロワーモデル	150 ミリ秒間 2 番のドロワーを駆動	c	63
ドロワーモデル	200 ミリ秒間 2 番のドロワーを駆動	d	64
ドロワーモデル	250 ミリ秒間 2 番のドロワーを駆動	e	65
全モデル	水平タブ（HT）	5	35
全モデル	改行（LF）	6	36
全モデル	印字（CR）	7	37
全カッターモデル	フルカット ※	F	46
全カッターモデル	パーシャルカット ※	P	50
全モデル	NV メモリー登録された 1 番のデータを通常印字	G	47
全モデル	NV メモリー登録された 2 番のデータを通常印字	H	48
全モデル	NV メモリー登録された 3 番のデータを通常印字	I	49
全モデル	NV メモリー登録された 4 番のデータを通常印字	J	4A
全モデル	NV メモリー登録された 5 番のデータを通常印字	K	4B
全モデル	NV メモリー登録された 1 番のデータを横倍印字	Q	51
全モデル	NV メモリー登録された 2 番のデータを横倍印字	R	52
全モデル	NV メモリー登録された 3 番のデータを横倍印字	S	53
全モデル	NV メモリー登録された 4 番のデータを横倍印字	T	54
全モデル	NV メモリー登録された 5 番のデータを横倍印字	U	55
全モデル	NV メモリー登録された 1 番のデータを縦倍印字	V	56
全モデル	NV メモリー登録された 2 番のデータを縦倍印字	W	57
全モデル	NV メモリー登録された 3 番のデータを縦倍印字	X	58
全モデル	NV メモリー登録された 4 番のデータを縦倍印字	Y	59
全モデル	NV メモリー登録された 5 番のデータを縦倍印字	Z	5A
全モデル	NV メモリー登録された 1 番のデータを 4 倍印字	[	5B
全モデル	NV メモリー登録された 2 番のデータを 4 倍印字	]	5D
全モデル	NV メモリー登録された 3 番のデータを 4 倍印字	^	5E
全モデル	NV メモリー登録された 4 番のデータを 4 倍印字	_	5F
全モデル	NV メモリー登録された 5 番のデータを 4 倍印字	`	60

※プリンターのメモリースイッチの設定により、指定通りのカット動作にならない事があります。

・プリンターがサポートしていない機能は働きません。

10. 用紙サイズ

ドライバーをインストールすると、各プリンターがサポートする用紙幅、印字桁数と下記の用紙長とを組み合わせた用紙サイズが登録されます。

A) 用紙長

	用紙長
a)	A4 長（297mm）
b)	B5 長（257mm）
c)	6 in Page（6 in）
d)	Letter length（11 in）
e)	Executive length（10.5 in）
f)	Receipt（3276 mm）
g)	ユーザー設定

B) 用紙幅

モデル	用紙幅
CT-S251	58mm(32 桁) 58mm(36 桁)
CT-S280/281	58mm（32 桁） 58mm（30 桁）
CT-P290/291	58mm（36 桁）
CT-S300/310 PMU2xxx PPU-700	80mm（48 桁） 80mm（42 桁） 58mm（32 桁） 58mm（30 桁）
CT-P292/293	80mm（48 桁） 58mm（36 桁）
CBM1000II BD2-xxxx	80mm（48 桁） 80mm（42 桁） 58mm（36 桁） 58mm（30 桁）
CT-S601(II) CT-S651(II) CT-S801(II) CT-S851(II) CT-S2000	83mm（53 桁） 80mm（48 桁） 80mm（42 桁） 58/60mm（36 桁） 58mm（30 桁）

CT-S4000	112mm（69 桁） 83mm（53 桁） 80mm（48 桁） 80mm（42 桁）
CD-S500 series	76mm 69.5mm 58.5mm

・前述の章「Paper Type」が Receipt に設定されている場合、選んだ用紙サイズよりも印字サイズが少ない場合は、最終行の直後で用紙カット、または印字停止します。Ticket に設定されている場合は選んだ用紙サイズ分の排出が行われます。

・MS-Word や MS-ACCESS 等のアプリケーションからは f) Receipt（3276 mm）の用紙は選択出来ません。

・ユーザー設定用紙は以下の方法で定義出来ます。定義出来る用紙サイズはプリンタードライバーの機種によって異なります。

Windows 7

「デバイスとプリンター」ウィンドウを開き、何か一つのプリンターを選択します。上部メニューの右側の部分に「プリントサーバーのプロパティ」のボタンが表示されますので、これをクリックします。

その他 OS。

プリンタとFaxの Window のメニューから

ファイル → サーバーのプロパティ

右のように「プリントサーバーのプロパティ」が表示されますので、「用紙タブ」より「新しい用紙を作成する」にチェックを入れ、「用紙名」、下方の用紙サイズを定義した後に、「用紙の保存」を押して作成します。

・ 定義出来るユーザー定義用紙のサイズは以下のようになります。

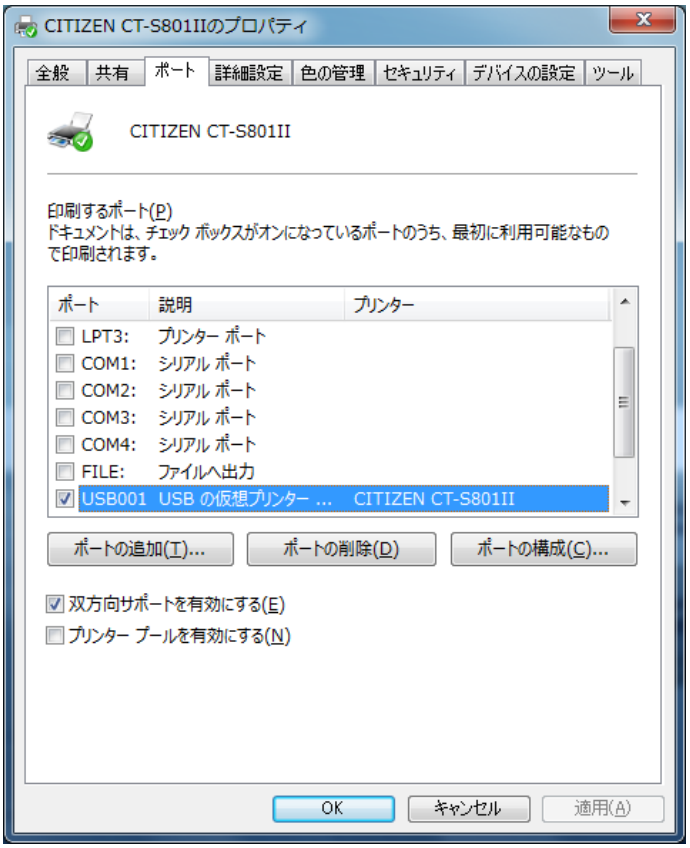
ドライバー名	ユーザー定義用紙範囲	
	最小	最大
CT-S4000	(20,20)	(896,26182):但し最大印字幅 832
CT-S4000 Label	(203, 203)	(896,26182):但し最大印字幅 832
CT-S601(II) CT-S651(II) CT-S801(II) CT-S851(II) CT-S2000 CT-P292/293	(20,20)	(664,26182):但し最大印字幅 640
CT-S651(II) Black Mark CT-S801 Label CT-S851(II) Black Mark CT-S2000 Label	(203, 203)	(664,26182):但し最大印字幅 640
PPU-700	(203, 719)	(640,26182):但し最大印字幅 575
CBM1000II CT-S300 CT-S310 PMU2xxx BD2-222x	(20,20)	(640,26182):但し最大印字幅 575
CT-S300 Black Mark CT-S310 Black Mark CBM1000II Label PMU23xxIII Presenter	(203, 203)	(640,26182):但し最大印字幅 575
CT-P290/291	(20, 20)	(480, 26182):但し最大印字幅 407
CT-S280 CT-S281 BD2-428x	(20,20)	(464,26182):但し最大印字幅 383
CT-S281 Label	(203, 203)	(464,26182):但し最大印字幅 383
CT-S251	(20, 20)	(464, 26182):但し最大印字幅 431
CD-S50x	(144, 144)	(432, 18572)但し最大印字幅 400

※単位はドット。1 ドットは 1/203 インチ換算。

11. プリンターステータス

11. 1 プリンターステータスの取得

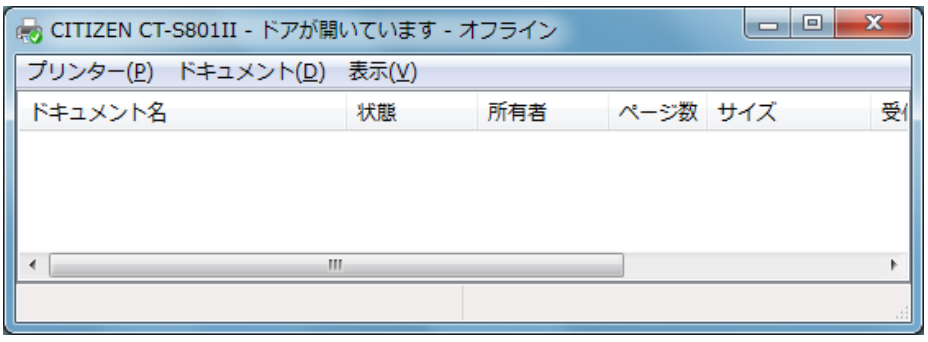
本プリンタードライバーは独自のランゲージモニター（ステータスモニター）により双方向通信を実現することで、既存の Windows の機能を使ってプリンターの状態を取得する事を可能としています。プリンタードライバーのプロパティを開いて、「双方向サポートを有効にする」にチェックが入っている事を確認して下さい。



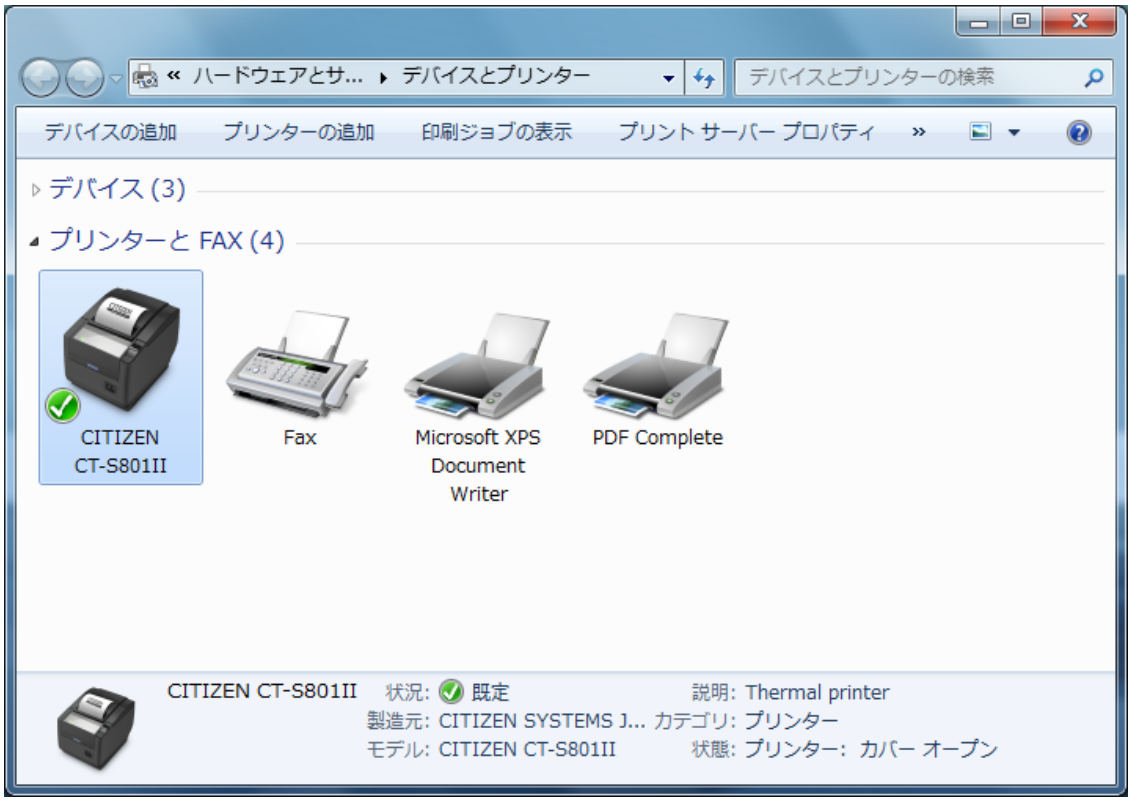
・プリンターのメモリースイッチ設定にある「Busy 条件」設定を、「バッファフル」設定に変更してご使用下さい。

プリンターの状態が変化すると、スプーラーウィンドウの上部、またはドライバーリストの「状態」桁部に、直ちに内容が表示されます。

（スプーラー）



（ドライバーリスト）



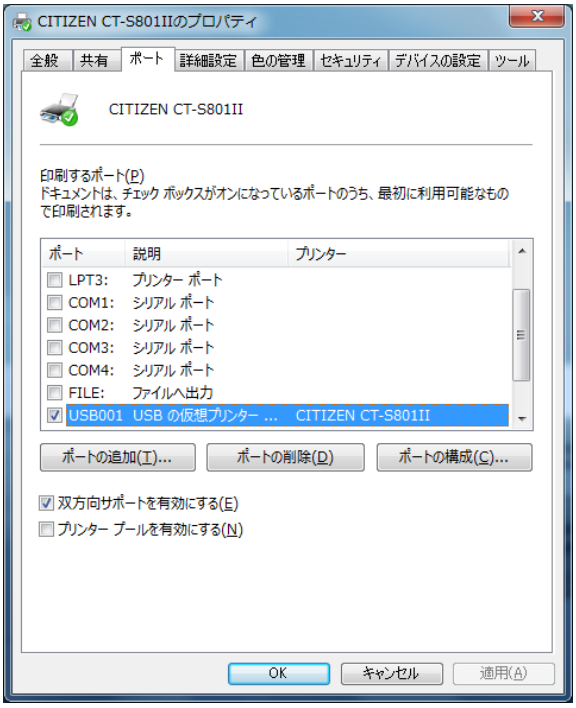
プリンター状態と表示される内容は以下の通りになります。

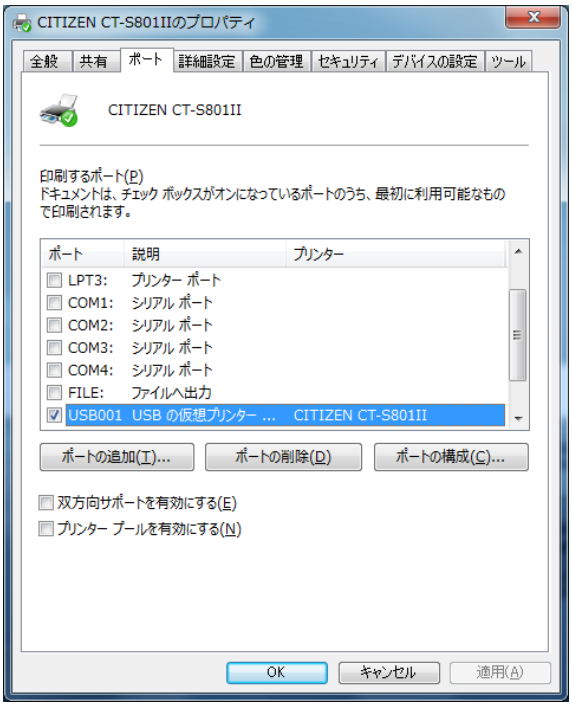
プリンター状態	スプーラー表示	ドライバーリスト表示
カバーオープン状態	ドアが開いています	カバーオープン
ペーバーエンド状態	用紙切れ	用紙切れ
ペーパーニアエンド状態	用紙の問題	用紙エラー
カッターエラー状態	ユーザー手順操作	要調査
印字ヘッド高温状態	ユーザー手順操作	要調査
FEED キー紙送り中状態	ユーザー手順操作	要調査
ケーブル抜け状態	オフライン	オフライン
プリンター電源 OFF 状態	オフライン	オフライン
プリンターオフライン状態	オフライン	オフライン

上記の状態が同時に複数検出されると、組合されて表示されます。「カバーオープン状態」、「ペーパーエンド状態」、「カッターエラー状態」、「印字ヘッド高温状態」、「FEED キー紙送り状態」、同時に”オフライン”の表示もされます。

11．2 双方向通信の有効／無効

プリンターと PC が接続された状態で、プリンターの電源を ON にします。プリンタードライバーのプロパティを開いて、「双方向サポートを有効にする」にチェックを入れる事で双方通信が有効、チェックを外す事で双方向通信が無効となります。





設定変更後、必ず印刷スプーラー(Print Spooler)を再起動して下さい。コマンドプロンプトを起動（Windows Vista 以降の OS をご使用の場合、必ずマウス右クリックメニューの「管理者として実行」から起動して下さい）して、下記コマンドを入力する事で印刷スプーラーをリセット出来ます。

```
net stop spooler        ← 印刷スプーラーの停止
net start spooler       ← 印刷スプーラーの開始
```

12. 特定アプリケーションにおける使用例

12. 1 Microsoft Word での使用例

以下は Microsoft Word 2003 でのプリンターフォントを利用する印字の例です。

A. メニューから、

「ファイル」－「印刷」－「プリンタ名」

で使うプリンターを選択し、「閉じる」を押します。

B. 次に、メニューから、

「ファイル」－「ページ設定」－「用紙」

と進み、「用紙サイズ」を選びます。

C. そしてその隣の「余白」でマージンを設定します。余白を 0 にすると、「余白が印刷できない領域に設定されています」とメッセージが出て、そのまま「修正」を押すと自動的に最小のマージンに設定されます。

D. フォントリストからプリンターフォントを選びます。（前述の章「プリンターフォント」を参照下さい）

E. フォントサイズリストでフォントサイズを選びます。（前述の章「プリンターフォント」を参照下さい）

F. Word 画面上に印刷させるデータを入力します。

ここから先が重要です。

G. メニューから、

「ツール」－「オプション」－「互換性」

と進み、対象となるアプリケーションを「Microdoft Word 6.9/95」にするか、またはオプション内にある「文書をレイアウトするときにプリンターの設定に従う」をチェックして下さい。

H. 最後に、

「ファイル」－「印刷」－「OK」

で印刷されます。

特殊機能（ドロワー1 を開ける）を使う場合の例ですが、ステップA～Cは上記と同じです。

D. フォントリストから「Control」を選びます。

E. フォントサイズリストで「12」を選びます。

F. Word の画面上に、"A"と入力します。（50ms のパルスをドロワー1 に送る機能は A に割り当てられているため）

ステップG、Hを行い、印刷します。

12. 2 Visual Basic でのプログラム例

以下は Visual Basic 6.0 にて「CITIZEN CT-S2000」のドライバーで各種印字、特殊機能を使う例です。

```
'CITIZEN CT-S2000 をデフォルトのプリンターに設定する
Dim X As Printer
For Each X In Printers
      If X.DeviceName = “CITIZEN CT-S2000” Then
            Set Printer = X
            Exit For
      End If
Next

‘TrueTypeフォントで印字
Printer.FontSize=10
Printer.FontName=”Arial”
Printer.Print “Font Arial / Size 10”

‘プリンターフォントで印字
Pritner.FontSize=12
Printer.FontName=”15 cpi”
Printer.Print “15 cpi / Size 12”

‘バーコードの印字
Printer.FontSize=42
Printer.FontName=”Code39”
Pritner.Print “ABC123456”

‘ドロワーを開く
Printer.FontSize=12
Pritner.FontName=”Control”
Printer.Print “A”    ‘Drawer 1 at 50ms

‘カット
Printer.FontSize=12
Pritner.FontName=”Control”
Printer.print “P”    ‘Partial cut

‘NV メモリーの一番目に登録されているイメージデータを印字する
Printer.FontSize=12
Pritner.FontName=”Control”
Printer.print “G”    ‘Print #1 gaphic data in NV memory

Printer.EndDoc


“xx cpi”フォントで、プリンターコマンドをドライバー経由でおくることもできます。
（この方法は推奨いたしません。ご自身のリスクでお使い下さい）
例

Pritner.FontSize=12
Printer.FontName=”15 cpi”
Printer.Print CHR$(&H1B) + “V” + CHR$(1)    ‘90 degrees right turned.
Printer.Print “15 cpi / Size 12”
Printer.EndDoc
```

12. 3 Visual C++でのプログラミング例

以下は Visual C++ 6.0 にて MFC ライブラリを使用した印刷例です。

```
//使用するプリンタの DEVMODE 構造体を取得する
//lpdevmode は確保した DEVMODE 構造体へのポインタ

//指定されたプリンタのデバイスコンテキストを作成します
HDC hdc;
hdc = CreateDC(_T("CITIZEN CT-S2000"), _T("CITIZEN CT-S2000"), NULL, lpdevmode);
if (!hdc) return;

// プリンターのデバイスコンテキストを CDC へアタッチする
CDC PrinterCDC;
PrinterCDC.Attach(hdc);

// プリントジョブを作成する
DOCINFO docInfo;
memset(&docInfo, 0, sizeof(docInfo));
docInfo.cbSize = sizeof(docInfo);
docInfo.lpszDocName = _T("Driver Print Test"); // スプーラーに表示される文書名
docInfo.lpszOutput = NULL;
docInfo.lpszDatatype = NULL;
docInfo.fwType = NULL;

// 印刷開始
int ypos = 0;
PrinterCDC.StartDoc(&docInfo);
PrinterCDC.StartPage();

// TrueType フォントで印字
CFont setFont;
CFont* poldFont;
CSize setSize;
CString txtPrintData;
setFont.CreatePointFont(10*10, _T("Arial"), &PrinterCDC);
poldFont = PrinterCDC.SelectObject(&setFont);
txtPrintData = _T("Font Arial / Size 10");
setSize = PrinterCDC.GetTextExtent(txtPrintData);
PrinterCDC.TextOut(0, ypos, txtPrintData);
PrinterCDC.SelectObject(poldFont);
setFont.DeleteObject();
ypos += setSize.cy;

// プリンターフォントでの印字
setFont.CreatePointFont(12*10, _T("15 cpi"), &PrinterCDC);
poldFont = PrinterCDC.SelectObject(&setFont);
txtPrintData = _T("15 cpi / Size 12");
setSize = PrinterCDC.GetTextExtent(txtPrintData);
PrinterCDC.TextOut(0, ypos, txtPrintData);
PrinterCDC.SelectObject(poldFont);
setFont.DeleteObject();
ypos += setSize.cy;

// バーコードの印字
setFont.CreatePointFont(42*10, _T("Code39"), &PrinterCDC);
```

```
poldFont = PrinterCDC.SelectObject(&setFont);
txtPrintData = _T("ABC123456");
setSize = PrinterCDC.GetTextExtent(txtPrintData);
PrinterCDC.TextOut(0, ypos, txtPrintData);
PrinterCDC.SelectObject(poldFont);
setFont.DeleteObject();
ypos += setSize.cy;

// ドロワーを開く
setFont.CreatePointFont(12*10, _T("Control"), &PrinterCDC);
poldFont = PrinterCDC.SelectObject(&setFont);
txtPrintData = _T("A");
setSize = PrinterCDC.GetTextExtent(txtPrintData);
PrinterCDC.TextOut(0, ypos, txtPrintData);
PrinterCDC.SelectObject(poldFont);
setFont.DeleteObject();
ypos += setSize.cy;

// カット
setFont.CreatePointFont(12*10, _T("Control"), &PrinterCDC);
poldFont = PrinterCDC.SelectObject(&setFont);
txtPrintData = _T("P");
setSize = PrinterCDC.GetTextExtent(txtPrintData);
PrinterCDC.TextOut(0, ypos, txtPrintData);
PrinterCDC.SelectObject(poldFont);
setFont.DeleteObject();
ypos += setSize.cy;

// NV メモリーの一番目に登録されているイメージデータを印字する
setFont.CreatePointFont(12*10, _T("Control"), &PrinterCDC);
poldFont = PrinterCDC.SelectObject(&setFont);
txtPrintData = _T("G");
setSize = PrinterCDC.GetTextExtent(txtPrintData);
PrinterCDC.TextOut(0, ypos, txtPrintData);
PrinterCDC.SelectObject(poldFont);
setFont.DeleteObject();
ypos += setSize.cy;

// 印刷終了
PrinterCDC.EndPage();
PrinterCDC.EndDoc();

// デバイスコンテキストの開放
PrinterCDC.Detach();
DeleteDC(PrinterHDC);
```

12. 4 Visual Basic.Net でのプログラミング例

以下は Visual Basic .Net にて Win32API を用いた印刷例です。

```
'Imports System.Runtime.InteropServices


Public Class PrinterControl
    <StructLayout(LayoutKind.Sequential)> _
    Public Structure DOCINFO
        Public cbSize As Integer
        Public lpszDocName As String
        Public lpszOutput As String
        Public lpszDatatype As String
        Public fwType As Integer
    End Structure

    <DllImport("winspool.drv", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function OpenPrinter(pPrinterName As [String], ByRef phPrinter As
IntPtr, pDefault As IntPtr) As Long
    End Function

    <DllImport("winspool.drv", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function ClosePrinter(hPrinter As IntPtr) As Long
    End Function

    <DllImport("winspool.drv", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function DocumentProperties(hwnd As IntPtr, hPrinter As IntPtr,
<MarshalAs(UnmanagedType.LPWStr)> pDeviceName As String, pDevModeOutput As
IntPtr, pDevModeInput As IntPtr, fMode As Integer) As Integer
    End Function

    Private Const DM_OUT_BUFFER As Integer = &H2
    Private Const DM_PROMPT As Integer = &H4
    Private Const DM_IN_PROMPT As Integer = DM_PROMPT
    Private Const DM_IN_BUFFER As Integer = &H8

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function CreateDC(lpszDriver As String, lpszDevice As String,
lpszOutput As String, lpInitData As IntPtr) As IntPtr
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function DeleteDC(hDC As IntPtr) As Boolean
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function StartDoc(hdc As IntPtr,
<MarshalAs(UnmanagedType.Struct)> ByRef lpdi As DOCINFO) As Int32
    End Function
```

```
    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function EndDoc(hdc As IntPtr) As Int32
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function StartPage(hDC As IntPtr) As Int32
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function EndPage(hDC As IntPtr) As Int32
    End Function

    <StructLayout(LayoutKind.Sequential)> _
    Public Class LOGFONT
        Public Const LF_FACESIZE As Integer = 32
        Public lfHeight As Integer
        Public lfWidth As Integer
        Public lfEscapement As Integer
        Public lfOrientation As Integer
        Public lfWeight As Integer
        Public lfItalic As Byte
        Public lfUnderline As Byte
        Public lfStrikeOut As Byte
        Public lfCharSet As Byte
        Public lfOutPrecision As Byte
        Public lfClipPrecision As Byte
        Public lfQuality As Byte
        Public lfPitchAndFamily As Byte
        <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=LF_FACESIZE)> _
        Public lfFaceName As String
    End Class

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function CreateFontIndirect(<[In](),
MarshalAs(UnmanagedType.LPStruct)> lplf As LOGFONT) As IntPtr
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function SelectObject(hdc As IntPtr, handle As IntPtr) As IntPtr
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function DeleteObject(handle As IntPtr) As Boolean
    End Function

    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function TextOut(hdc As IntPtr, nXStart As Integer, nYStart As
Integer, lpString As String, cbString As Integer) As Integer
    End Function
```

```
    <DllImport("gdi32.dll", CharSet:=CharSet.Ansi, ExactSpelling:=False,
CallingConvention:=CallingConvention.StdCall)> _
    Public Shared Function GetDeviceCaps(hdc As IntPtr, nIndex As Integer) As Integer
    End Function

    Private Const LOGPIXELSY As Integer = 90


    Public Shared Sub PrinterFontTest(printerName As [String])
        Dim hPrinter As IntPtr = IntPtr.Zero
        OpenPrinter(printerName, hPrinter, IntPtr.Zero)

        ' プリンターデバイスコンテキストの作成
        Dim hwnd As IntPtr = IntPtr.Zero
        Dim size As Integer = DocumentProperties(hwnd, hPrinter, printerName,
IntPtr.Zero, IntPtr.Zero, 0)
        Dim pDevmode As IntPtr = Marshal.AllocHGlobal(size)
        Dim ret As Integer = DocumentProperties(hwnd, hPrinter, printerName,
pDevmode, IntPtr.Zero, DM_OUT_BUFFER)
        Dim hDC As IntPtr = CreateDC(Nothing, printerName, Nothing, pDevmode)

        ' プリントジョブを作成する
        Dim di As New DOCINFO()
        di.cbSize = Marshal.SizeOf(di)
        di.lpszDocName = "Driver Test Print"

        ' 印刷開始
        StartDoc(hDC, di)
        StartPage(hDC)

        Dim lf As New LOGFONT()
        Dim hFont As IntPtr = IntPtr.Zero
        Dim oldFont As IntPtr = IntPtr.Zero
        Dim str As String
        Dim fontsize As Integer, pointsize As Integer

        ' TrueType フォントで印字
        pointsize = 15
        fontsize = CInt((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) ¥ 720)
        lf.lfHeight = -fontsize
        lf.lfFaceName = "Arial"
        hFont = CreateFontIndirect(lf)
        oldFont = SelectObject(hDC, hFont)
        str = "Font Arial / Size 15"
        TextOut(hDC, 10, 10, str, str.Length)
        SelectObject(hDC, oldFont)
        DeleteObject(hFont)

        ' プリンターフォントで印字
        pointsize = 12
        fontsize = CInt((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) ¥ 720)
        lf.lfHeight = -fontsize
        lf.lfFaceName = "15 cpi"
        hFont = CreateFontIndirect(lf)
        oldFont = SelectObject(hDC, hFont)
```

```
str = "15 cpi / Size 12"
TextOut(hDC, 10, 50, str, str.Length)
SelectObject(hDC, oldFont)
DeleteObject(hFont)

' バーコードの印字
pointsize = 42
fontsize = CInt((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) ¥ 720)
lf.lfHeight = -fontsize
lf.lfFaceName = "Code39"
hFont = CreateFontIndirect(lf)
oldFont = SelectObject(hDC, hFont)
str = "ABC123456"
TextOut(hDC, 10, 90, str, str.Length)
SelectObject(hDC, oldFont)
DeleteObject(hFont)

' ドロワーを開く
pointsize = 12
fontsize = CInt((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) ¥ 720)
lf.lfHeight = -fontsize
lf.lfFaceName = "Control"
hFont = CreateFontIndirect(lf)
oldFont = SelectObject(hDC, hFont)
str = "A"
TextOut(hDC, 10, 130, str, str.Length)
SelectObject(hDC, oldFont)
DeleteObject(hFont)

' カット
pointsize = 12
fontsize = CInt((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) ¥ 720)
lf.lfHeight = -fontsize
lf.lfFaceName = "Control"
hFont = CreateFontIndirect(lf)
oldFont = SelectObject(hDC, hFont)
str = "P"
TextOut(hDC, 10, 170, str, str.Length)
SelectObject(hDC, oldFont)
DeleteObject(hFont)

' NV メモリーの一番目に登録されているイメージデータを印字する
pointsize = 12
fontsize = CInt((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) ¥ 720)
lf.lfHeight = -fontsize
lf.lfFaceName = "Control"
hFont = CreateFontIndirect(lf)
oldFont = SelectObject(hDC, hFont)
str = "G"
TextOut(hDC, 10, 210, str, str.Length)
SelectObject(hDC, oldFont)
DeleteObject(hFont)

' 印刷終了
EndPage(hDC)
EndDoc(hDC)
```

```
        ' プリンターデバイスコンテキストの解放
        DeleteDC(hDC)
        Marshal.FreeHGlobal(pDevmode)
        ClosePrinter(hPrinter)
    End Sub
End Class
```

12．5 Visual C#.Net でのプログラミング例

以下は Visual C#.Net にて Win32API を用いた印刷例です。

```
using System.Runtime.InteropServices;


    public class PrinterControl
    {
        [StructLayout(LayoutKind.Sequential)]
        public struct DOCINFO
        {
            public int cbSize;
            public string lpszDocName;
            public string lpszOutput;
            public string lpszDatatype;
            public int fwType;
        }

        [DllImport("winspool.drv", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern long OpenPrinter(String pPrinterName, ref IntPtr phPrinter,
IntPtr pDefault);

        [DllImport("winspool.drv", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern long ClosePrinter(IntPtr hPrinter);

        [DllImport("winspool.drv", CharSet = CharSet.Ansi,  ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern int DocumentProperties(IntPtr hwnd, IntPtr hPrinter,
[MarshalAs(UnmanagedType.LPWStr)] string pDeviceName, IntPtr pDevModeOutput,
IntPtr pDevModeInput, int fMode);

            private const int DM_OUT_BUFFER = 0x2;
            private const int DM_PROMPT = 0x4;
            private const int DM_IN_PROMPT = DM_PROMPT;
            private const int DM_IN_BUFFER = 0x8;

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern IntPtr CreateDC(string lpszDriver, string lpszDevice, string
lpszOutput, IntPtr lpInitData);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern bool DeleteDC(IntPtr hDC);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern Int32 StartDoc(IntPtr hdc,
[MarshalAs(UnmanagedType.Struct)] ref DOCINFO lpdi);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern Int32 EndDoc(IntPtr hdc);
```

```
        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern Int32 StartPage(IntPtr hDC);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern Int32 EndPage(IntPtr hDC);

        [StructLayout(LayoutKind.Sequential)]
        public class LOGFONT
        {
            public const int LF_FACESIZE = 32;
            public int lfHeight;
            public int lfWidth;
            public int lfEscapement;
            public int lfOrientation;
            public int lfWeight;
            public byte lfItalic;
            public byte lfUnderline;
            public byte lfStrikeOut;
            public byte lfCharSet;
            public byte lfOutPrecision;
            public byte lfClipPrecision;
            public byte lfQuality;
            public byte lfPitchAndFamily;
            [MarshalAs(UnmanagedType.ByValTStr, SizeConst = LF_FACESIZE)]
            public string lfFaceName;
        }

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern IntPtr CreateFontIndirect([In,
MarshalAs(UnmanagedType.LPStruct)] LOGFONT lplf);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern IntPtr SelectObject(IntPtr hdc, IntPtr handle);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern bool DeleteObject(IntPtr handle);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern int TextOut(IntPtr hdc, int nXStart, int nYStart, string
lpString, int cbString);

        [DllImport("gdi32.dll", CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
        public static extern int GetDeviceCaps(IntPtr hdc, int nIndex);

            private const int LOGPIXELSY = 90;


        public static void PrinterFontTest(String printerName)
        {
```

```
            IntPtr hPrinter = IntPtr.Zero;
            OpenPrinter(printerName, ref hPrinter, IntPtr.Zero);

            // プリンターデバイスコンテキストの作成
            IntPtr hwnd = IntPtr.Zero;
            int size = DocumentProperties(hwnd, hPrinter, printerName, IntPtr.Zero,
IntPtr.Zero, 0);
            IntPtr pDevmode = Marshal.AllocHGlobal(size);
            int ret = DocumentProperties(hwnd, hPrinter, printerName, pDevmode,
IntPtr.Zero, DM_OUT_BUFFER);
            IntPtr hDC = CreateDC(null, printerName, null, pDevmode);

            // プリントジョブを作成する
            DOCINFO di = new DOCINFO();
            di.cbSize = Marshal.SizeOf(di);
            di.lpszDocName = "Driver Test Print";

            // 印刷開始
            StartDoc(hDC, ref di);
            StartPage(hDC);

            LOGFONT lf = new LOGFONT();
            IntPtr hFont = IntPtr.Zero;
            IntPtr oldFont = IntPtr.Zero;
            string str;
            int fontsize, pointsize;

            // TrueType フォントで印字
            pointsize = 15;
            fontsize = (int)((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) / 720);
            lf.lfHeight = -fontsize;
            lf.lfFaceName = "Arial";
            hFont = CreateFontIndirect(lf);
            oldFont = SelectObject(hDC, hFont);
            str = "Font Arial / Size 15";
            TextOut(hDC, 10, 10, str, str.Length);
            SelectObject(hDC, oldFont);
            DeleteObject(hFont);

            // プリンターフォントで印字
            pointsize = 12;
            fontsize = (int)((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) / 720);
            lf.lfHeight = -fontsize;
            lf.lfFaceName = "15 cpi";
            hFont = CreateFontIndirect(lf);
            oldFont = SelectObject(hDC, hFont);
            str = "15 cpi / Size 12";
            TextOut(hDC, 10, 50, str, str.Length);
            SelectObject(hDC, oldFont);
            DeleteObject(hFont);

            // バーコードの印字
            pointsize = 42;
            fontsize = (int)((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) / 720);
            lf.lfHeight = -fontsize;
            lf.lfFaceName = "Code39";
```

```
            hFont = CreateFontIndirect(lf);
            oldFont = SelectObject(hDC, hFont);
            str = "ABC123456";
            TextOut(hDC, 10, 90, str, str.Length);
            SelectObject(hDC, oldFont);
            DeleteObject(hFont);

            // ドロワーを開く
            pointsize = 12;
            fontsize = (int)((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) / 720);
            lf.lfHeight = -fontsize;
            lf.lfFaceName = "Control";
            hFont = CreateFontIndirect(lf);
            oldFont = SelectObject(hDC, hFont);
            str = "A";
            TextOut(hDC, 10, 130, str, str.Length);
            SelectObject(hDC, oldFont);
            DeleteObject(hFont);

            // カット
            pointsize = 12;
            fontsize = (int)((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) / 720);
            lf.lfHeight = -fontsize;
            lf.lfFaceName = "Control";
            hFont = CreateFontIndirect(lf);
            oldFont = SelectObject(hDC, hFont);
            str = "P";
            TextOut(hDC, 10, 170, str, str.Length);
            SelectObject(hDC, oldFont);
            DeleteObject(hFont);

            // NV メモリーの一番目に登録されているイメージデータを印字する
            pointsize = 12;
            fontsize = (int)((pointsize * 10) * GetDeviceCaps(hDC, LOGPIXELSY) / 720);
            lf.lfHeight = -fontsize;
            lf.lfFaceName = "Control";
            hFont = CreateFontIndirect(lf);
            oldFont = SelectObject(hDC, hFont);
            str = "G";
            TextOut(hDC, 10, 210, str, str.Length);
            SelectObject(hDC, oldFont);
            DeleteObject(hFont);

            // 印刷終了
            EndPage(hDC);
            EndDoc(hDC);

            // プリンターデバイスコンテキストの解放
            DeleteDC(hDC);
            Marshal.FreeHGlobal(pDevmode);
            ClosePrinter(hPrinter);
        }
    }
```